

---

# Supplementary Material for “MineAnyBuild: Benchmarking Spatial Planning for Open-world AI Agents”

---

Project Website: <https://mineanybuild.github.io/>

## Contents

<b>A</b>	<b>Broader Impacts, Limitations, and Future Directions</b>	<b>3</b>
A.1	Boarder Impacts . . . . .	3
A.2	Limitations . . . . .	3
A.3	Future Directions . . . . .	4
<b>B</b>	<b>Minecraft</b>	<b>5</b>
B.1	Environment Details of Minecraft Game . . . . .	5
B.2	Simulator Environments . . . . .	5
B.3	Tools for Evaluation and Visualization . . . . .	5
B.3.1	Mineflayer Viewer . . . . .	6
B.3.2	Replay Mod . . . . .	7
<b>C</b>	<b>Datasheets</b>	<b>8</b>
C.1	Dataset Description . . . . .	8
C.2	License . . . . .	8
C.3	Format . . . . .	8
C.4	Data Field Descriptions . . . . .	9
C.4.1	Architectures . . . . .	9
C.4.2	Tasks . . . . .	10
C.5	Data Examples . . . . .	10
C.5.1	Architectures . . . . .	10
C.5.2	Tasks . . . . .	10
C.6	Croissant Metadata . . . . .	10
<b>D</b>	<b>Details for Benchmark, Tasks and Data Curation</b>	<b>11</b>
D.1	Details of Dataset . . . . .	11
D.2	Details of Tasks . . . . .	13

D.2.1	More Details of Executable Spatial Plan Generation Task . . . . .	13
D.2.2	More Details of Creativity Task . . . . .	13
D.2.3	More Details of Spatial Understanding Task . . . . .	14
D.2.4	More Details of Spatial Reasoning Task . . . . .	14
D.2.5	More Details of Spatial Commonsense Task . . . . .	14
D.3	Details of Data Curation Pipeline . . . . .	15
D.3.1	Details About Infinitely Expandable Paradigm . . . . .	15
D.3.2	Estimation Metric of Difficulty Factor . . . . .	16
D.3.3	Difficulty Tiers . . . . .	17
D.4	Discussion on Evaluation and Output Format . . . . .	17
D.4.1	Discussion on Evaluation and Output Format . . . . .	17
D.4.2	Details of Output Format . . . . .	18
<b>E</b>	<b>Prompts</b>	<b>18</b>
E.1	Task prompts . . . . .	18
E.2	Evaluation Prompts . . . . .	21
E.3	Data Curation Prompts . . . . .	24
<b>F</b>	<b>Additional Experimental Information, Results and Analyses</b>	<b>25</b>
F.1	Agents . . . . .	25
F.2	Compute Resources and Cost . . . . .	25
F.3	Evaluation Metrics . . . . .	25
F.3.1	Scores . . . . .	25
F.3.2	Formula for Comprehensive Score . . . . .	26
F.3.3	Explanations on “Overall” Score . . . . .	28
F.3.4	Error Bar Chart of Scoring by Critic Models . . . . .	28
F.4	Assessments for Reliability Evaluation . . . . .	28
F.5	Full Results of Output Success Rate . . . . .	29
F.6	Accuracy Evaluation Results on Spatial Reasoning Task . . . . .	29
F.7	Discussion on Human Evaluation . . . . .	29
F.8	Discussion on Possible Noise Brought by Critic Model . . . . .	30
F.9	Discussion on Abnormal Result on GPT-4o-mini in Spatial Reasoning Task . . . . .	30
F.10	Information and Metrics about Quality Control Processes in Data Curation Pipeline	30
F.11	Discussion on Other Baseline Evaluation . . . . .	30
F.12	Performance of Small-parameter Open-source MLLMs . . . . .	31
<b>G</b>	<b>Additional Visualization Results</b>	<b>31</b>
G.1	Executable Planning Output . . . . .	31
G.2	Failure Cases . . . . .	31
<b>H</b>	<b>Declaration of LLM Usage</b>	<b>33</b>

Our Supplementary Material for “MineAnyBuild: Benchmarking Spatial Planning for Open-world AI Agents” is organized as follows. In Section A, we discuss the impacts, limitations and future directions of our work. In Section B, we introduce the Minecraft game and its common simulated environments. In Section C, we present the datasheets of our datasets, including dataset description, data field descriptions, *etc.* In Section D, we provide the details for our benchmark, tasks, and data curation pipeline. In Section E, we present the prompts utilized for proprietary models and critic models in our benchmark. In Section F, we provide more experimental results, information and analyses. In Section G, we present more visualization results of our evaluation on AI agents. In Section H, we declare our usage of LLM in our paper and the Supplementary Material.

## A Broader Impacts, Limitations, and Future Directions

### A.1 Boarder Impacts

Spatial intelligence has become an important research dimension in developing open-world AI agents for handling various downstream applications. AI agents with spatial intelligence can better satisfy the need of spatial perception, understanding, memorization, *etc.*, in tasks like automatic assembly, architectural design, and robotic manipulation. Although existing open-world AI agents driven by Multi-modal Large Language Models (MLLMs) have exhibited astonishing capabilities in various text-based (1D) and image-based (2D) tasks, they still struggle to tackle tasks requiring spatial understanding and cognition from a 3D perspective. Moreover, existing benchmarks designed for evaluating spatial intelligence typically ignore the gap between abstract spatial understanding and actual task execution.

In light of this, we propose an innovative benchmark to evaluate a critical part in spatial intelligence, i.e., *spatial planning*, to bridge the gap between spatial reasoning and task completion. Our proposed **MineAnyBuild** benchmark standardizes task definitions, modules, and interfaces, enabling a comprehensive and rigorous evaluation of spatial planning capabilities for open-world AI agents, especially MLLM-based AI agents. Through our introduced infinitely expandable data collection paradigm, our benchmark can also acquire scalable data for facilitating spatial planning training and evaluation. We believe our proposed benchmark can make a significant stride in spatial intelligence research to promote the development of AI agents capable of spatial planning and reasoning, which brings great benefits for a wide range of real-world applications.

Creativity is also an important indicator for evaluating the aesthetics of 3D space tasks. Just like text creation in the 1D domain and image editing/painting creation in the 2D domain, with the rapid development of generative AI in recent years, the evaluation of aesthetic features and creative assessment under these dimensional data have always been one of the important current evaluation indicators. Similarly, we set up the creativity task to evaluate the agents’ assessment of the aesthetics and 3D potential of space tasks. Agents receive an instruction and are required to brainstorm block combinations for different parts of the architecture and outline a rough structure layout, to find ways to maximize creativity and the dynamic range of possible builds. Therefore, creativity not only reflects the cognitive depth of future AI systems, but also emerges as a novel and important criterion for agents towards Artificial General Intelligence (AGI) [1, 2].

Constrained by the limitations and biases of these AI agents, the potential negative impacts should also be considered when developing them to address real-world applications requiring spatial perception and cognition. For example, the uncertainty and hallucination of their outputs may cause unreliability and even safety problems, e.g., damage of public facilities. Therefore, besides the benchmark designed for encouraging AI agents research, it is also important to establish guidelines for accountability and design risk mitigation strategies when deploying these AI agents in real-world applications.

### A.2 Limitations

The main limitations of this work are summarized as follows:

- 1) Evaluation tools that we choose are relatively slow and require manual operations. We discuss the choice of tools in Section B.3.
- 2) Evaluation based on a scoring system still has drawbacks compared to evaluation based on models or fixed criteria. Although we believe that our scoring-based evaluation can better mimic the human

evaluation, it is obvious that this approach is somewhat inferior in terms of scientificity, reliability and stability when compared to some AI model-based evaluators.

3) The output format based on the 3D blueprint matrix is not the best so far. Although this is the best option we have chosen after careful consideration at present, this approach is not optimal due to the output token limits of MLLMs or the action sequence length of embodied agents. We also discuss it in Section D.4.

### A.3 Future Directions

Based on the limitations we discuss above, we outline the future directions to represent a roadmap for advancing the spatial intelligence research, especially the spatial planning, in this section.

- **Better Evaluation Tool:** Currently, the existing evaluation tools and frameworks are suitable for the previous tasks, such as skill learning and tech-tree, but they are not well-adapted to the building construction tasks that we proposed and lack a necessary interface for high-quality visualizations. We hope that more developers can cooperate to jointly develop and improve the evaluation tools for our proposed tasks of spatial planning in the future, so as to better promote the development of AI agents in spatial intelligence.
- **Advanced Evaluation System:** Designing evaluation methods beyond current scoring-based approach to obtain a more advanced evaluation system is crucial to evaluate AI agents and to improve the quality of agents’ spatial planning and intelligence.
- **Optimized Output Format:** Developing an output format which is easier for agents to understand and learn than the blueprint matrix or code generation is also beneficial, with which agents are able to “translate” their planning into executable results more effectively, reducing errors or biases caused by the output.
- **RL-based Agents:** On the basis of multiple existing simulator environments in Minecraft, implementing a basic RL-based agent is important at present. Training RL-based agents can greatly advance the development of Embodied AI research. We would like to develop RL-based agents in the future to better adapt to our MineAnyBuild benchmark and datasets.
- **Memory and Learning Modules:** We believe that the memory and learning modules are very important for the tasks and benchmarks we have proposed. Currently, most of our evaluations are based on zero-shot querying of MLLMs, and they usually respond with plans relying on their original knowledge base. If powerful memory modules are integrated, AI agents can gradually learn from small/simple architectures/assets, gain a deeper understanding of spatial planning step by step, and then complete more complex and large-scale buildings finally, akin to curriculum learning techniques, so as to build an ideal and perfect castle just like human players.
- **A General Agent Framework Combining MLLMs and RL Training:** Existing embodied AI frameworks mainly revolve around an MLLM-based “brain” and bottom-level modules for action execution, and such a framework can be extended to the tasks we proposed. In the future, a general agent framework that combines MLLMs with efficient RL training will greatly enhance the capabilities of agents to perform better on these planning tasks.
- **Transferring to Non-game Scenarios:** Our benchmark could be migrated to a similar non-game environment, such as Isaac Sim or AI2Thor simulators, for 3D scene generation tasks. This task requires a framework to retrieve assets from an asset library (e.g., Objaverse [3]) and properly place them into an indoor scene. How to ensure that the positions and rotations of all assets are correctly arranged is exactly a form of spatial planning. This task is one of the cutting-edge research directions in recent years, and the motivation of this is similar to that of our benchmark.
- **Transferring to Real-world Scenarios:** How to perform sim-to-real transfer for our benchmark is also one of our interests. We actually watched a video<sup>1</sup> on YouTube showing physical blocks with same appearance as those in the game (e.g., grass block), which can be spliced together using the built-in magnets, just like toy blocks and LEGO bricks. We plan to build a physical (real-world) version of our benchmark in the future, and study how

---

<sup>1</sup>Video address: <https://www.youtube.com/watch?v=UkyEyVoP8Jc>

agents trained in simulation can be deployed to manipulate these blocks in our real world to construct buildings.

## B Minecraft

Minecraft is a highly popular 3D sandbox video game developed by Microsoft’s Mojang Studios. The environment is primarily composed of 3D blocks, each distinguished by pixel-art textures on their surfaces that represent a variety of materials, such as dirt, stone, minerals, water, and trees. Minecraft facilitates a high degree of interaction with these blocks for both human players and AI agents, enabled through well-developed user interfaces or APIs. Players are capable of freely navigating the world, collecting these fundamental blocks, and subsequently positioning them on a standardized 3D integer-based coordinate grid to engage in diverse construction activities. Consequently, the diversity in block types, the extensive freedom of interaction, and the standardized 3D coordinate system establish Minecraft as an ideal environment for assessing the spatial reasoning and planning capabilities of large AI models.

Furthermore, Minecraft is supported by a vast and active user community that consistently generates an abundant and expanding collection of user-generated content. This content, including items such as modifications (mods), skins, texture packs, and custom maps, is readily available for download, a factor that underscores the platform’s significant extensibility.

### B.1 Environment Details of Minecraft Game

The Minecraft environment has several versions that are available for players to download and use. In our work, we recommend the version 1.16.5 and 1.20.4 for better adaptation to some simulator environments. We finally choose the version 1.20.4 for *mineflayer* simulator.

The Minecraft environment, specifically version 1.20.4, comprises an extensive set of 830+ standard block types. Each block type is uniquely identified by a textual name, which anyone can easily obtain the information on the website: <https://minecraft.wiki/w/Block>.

In Minecraft, players can do anything they can imagine. Players can build structures, craft tools, smelt ore, brew potions, trade with villagers and wandering traders, attack mobs, grow crops, raise animals in captivity, etc. Players even can use redstone to build a computer. This is a world of freedom and infinite possibilities.

The Minecraft world is divided into different areas called “biomes”. These biomes contain different blocks and plants and change how the land is shaped. There are 79 biomes in Minecraft 1.16.5, including ocean, plains, forest, desert, etc. Diverse environments have high requirements for the generalization of agents.

Current works [4, 5, 6, 7, 8, 9, 10, 11, 12] are merely confined to traditional embodied planning tasks like skill learning or tech-tree goals. Compared to these works, our benchmark MineAnyBuild is proposed to evaluate AI agents in spatial intelligence, which is an emerging research field regarding the ability of AI agents to reason about 3D space.

### B.2 Simulator Environments

We compile the simulator environments used in some current works in Table 1. It can be observed that most current works mainly focus on several categories such as MineRL [13], MineStudio [14], MineDojo [4] and Mineflayer [15].

Our benchmark MineAnyBuild is mainly built on the Mineflayer simulator environment. As we mentioned in Section A, we plan to expand and adapt our benchmark to these RL-based simulator environments in Table 1 as much as possible in the future.

### B.3 Tools for Evaluation and Visualization

We introduce two common tools for evaluation on our benchmark, and discuss our choice between them and the corresponding reasons.

Table 1: Various Minecraft agents in different code environments.

Code Environment	Agent	Publication	Task Type
MineRL [13]	VPT [6]	NeruIPS 2022	Skill Learning
	STEVE-1 [16]	NeruIPS 2023	Skill Learning
	Optimus-1 [8]	NeruIPS 2024	Skill Learning
	Optimus-2 [9]	CVPR 2025	Skill Learning
MineStudio [14]	Jarvis-1 [17]	T-PAMI 2024	Skill Learning
	GROOT-2 [18]	ICLR 2025	Skill Learning
	ROCKET-1 [12]	CVPR 2025	Skill Learning
MineDojo [4]	MineDojo [4]	NeruIPS 2022 D&B	Skill Learning & Architecture Building
	DEPS [19]	NeruIPS 2023	Skill Learning
	MP5 [20]	CVPR 2024	Skill Learning
Mineflayer [15]	Voyager [7]	TMLR 2024	Skill Learning & Tech Tree
	APT [21]	AAAI 2025 Workshop	Architecture Building

Compared with traditional Minecraft benchmarks and works, their evaluation is based on whether a task is successful or not, simply by checking data such as whether atomic actions match or environmental information and states match. However, our new benchmark MineAnyBuild focuses on evaluating the overall appearance of buildings and the spatial planning process, so it has high requirements for visual data. Therefore, we must rely on existing Minecraft visualization tools to visualize the buildings constructed by agents and then submit them to evaluators (critic models) for evaluation.

Currently, there are mainly two common visualization tools in relevant research works, namely Mineflayer Viewer and Replay Mod. We summarize the advantages and disadvantages of these two tools in Table 2.

Table 2: Advantages and disadvantages of two visulization tools.

Tools	Advantages	Disadvantages
Mineflayer Viewer	1) Easy to use. 2) Quick image acquisition speed.	1) Some blocks cannot be rendered, e.g., stairs and slabs. 2) Fixed perspective. 3) Poor customization.
Replay Mod	1) Any blocks can be correctly rendered. 2) Customizable perspectives. 3) High degree of freedom.	1) Slow image acquisition speed. 2) Non-automated. 3) Requiring manual operations.

Next, we will specifically introduce the characteristics and general usage of these two tools, and explain the reasons for our final choice (Replay Mod) in Section B.3.2.

### B.3.1 Mineflayer Viewer

**Mineflayer prismarine-viewer** is a tool proposed by PrismarineJS team. This tool provides Viewer and WorldView which make it possible to render a Minecraft world. This tool can be well integrated with Mineflayer [15] simulator environment to quickly generate the corresponding visual views from the agent’s current perspective. The Github repository of this tool is at <https://prismarinejs.github.io/prismarine-viewer/>.

Our initial plan for using this tool is as follows: We set up an observing agent (named *bot*) to observe the constructed objects from a specific position and at a specific angle, and then use this tool to take screenshots to obtain images from the corresponding perspectives and positions. This method has almost no delay and can be easily automated, making it highly suitable for evaluating agents.

However, we found that prismarine-viewer in Minecraft 1.16.5 and 1.20.4 can not render some important blocks that are frequently appeared in building tasks, such as *stairs* and *slabs*, as shown in

Figure 1. This poses a significant problem for building data and tasks involving structures such as houses, as it may result in these houses lacking well-designed roofs. Voyager [7] has answered an issue<sup>2</sup> that they finally didn’t choose this tool due to its poor-quality visualization. Therefore, we have to temporarily abandon this tool, and hoping that this tool can further address this rendering issue in the future.



Figure 1: Render error in Mineflayer prismarine-viewer.

While writing this Supplementary Material, we find that the project seems to be able to successfully render stairs in the latest version of Minecraft 1.21.4 (at <https://github.com/PrismarineJS/prismarine-viewer>). Therefore, we will update our benchmark in our GitHub code repository in the future to adapt to these latest versions, which will facilitate the further evaluation of AI agents.

### B.3.2 Replay Mod

**Replay Mod** is a mod for the popular sandbox game Minecraft which allows players to record, replay and share their gaming experience. It’s easy to use, but an incredibly powerful tool to create perfect Minecraft videos within minutes. The website of this tool is at <https://www.replaymod.com/>.

This evaluation tool can correctly render the blocks (e.g., *stairs* and *slabs*) that appear in architectural structures, and the quality of the output images/videos is relatively high. Although this tool requires some manual operations, these operations are relatively easy. Moreover, the generated images are less likely to cause errors that would lead to incorrect judgments of the agent’s output by the critic model.

Therefore, we finally selected this tool as our evaluation tool. We use this tool to obtain visualized videos, and extract frames from the videos through the *opencv-python* library to select the corresponding images as the input for the critic models, as shown in Figure 2. Specific operation tutorials can be referred to at <https://www.replaymod.com/docs/>. We will also provide corresponding reference documents and video processing code segments using *opencv-python* in our Github code repository in the future.

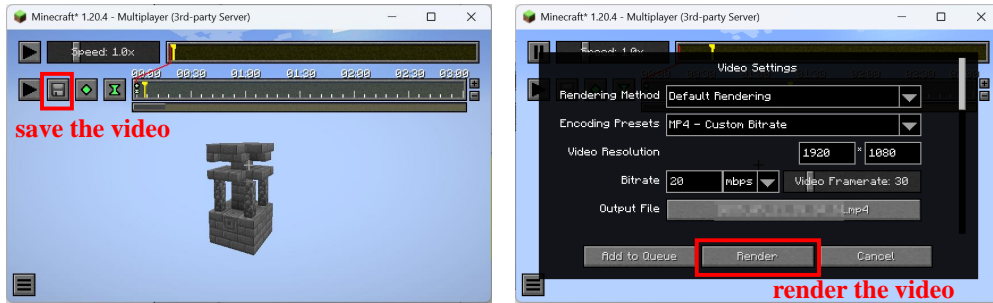


Figure 2: Visualization of Replay Mod.

<sup>2</sup><https://github.com/MineDojo/Voyager/issues/19>

## C Datasheets

### C.1 Dataset Description

Our dataset is built on several player-generated content (PGC) on the Internet. We conduct data curation process on this PGC and design various tasks based on the curated data. We recommend that any researchers using our dataset should use it for non-commercial AI research purposes only. We have released our dataset on Hugging Face: <https://huggingface.co/datasets/SaDil/MineAnyBuild>.

### C.2 License

Our MineAnyBuild benchmark is released under the CC BY-NC-SA 4.0 license. We also list the relevant licenses and terms-of-use of data resources utilized in the process of constructing our dataset as follow:

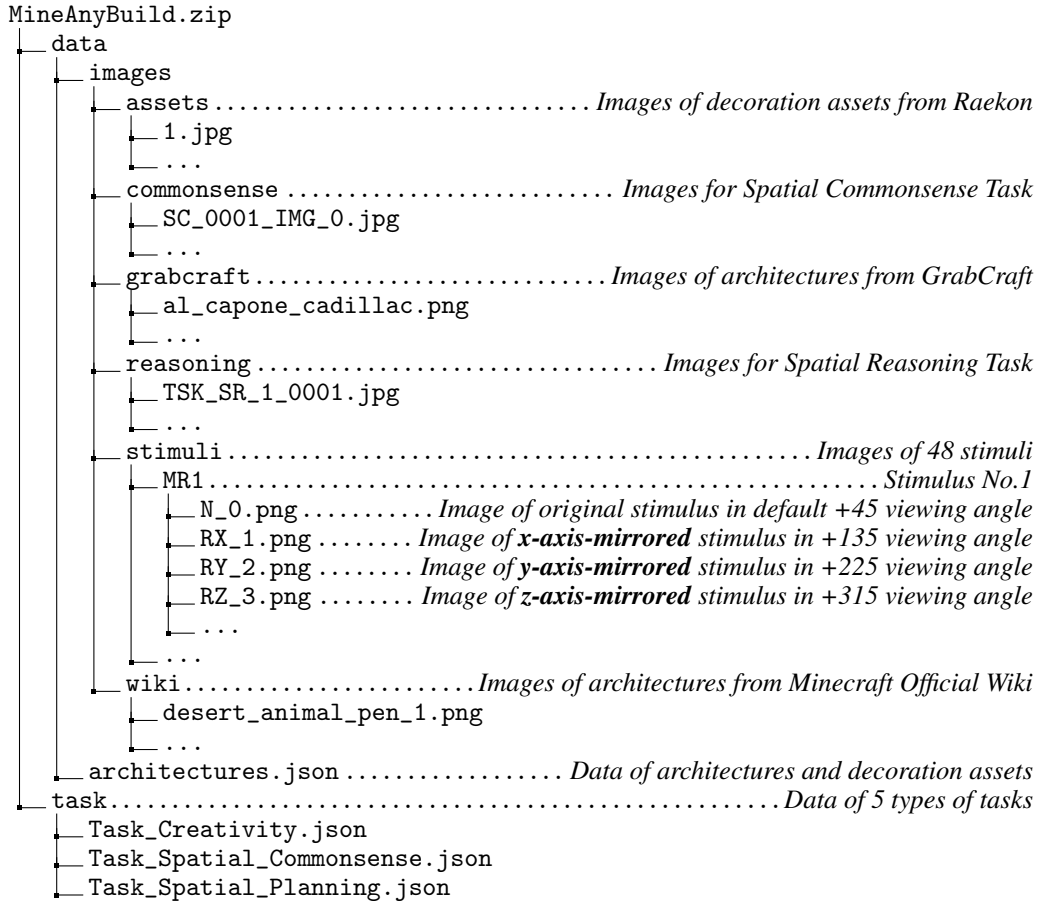
(1) **GrabCraft** [22]: CC BY-NC-SA 4.0 (Terms of Use).

(2) **Minecraft Official Wiki** [23]: CC BY-NC-SA 4.0 (Terms of Use).

(3) **Creations from Raekon**: CC BY-NC-SA 4.0 (Terms of Use). We may partially release our dataset to protect the copyright of this author, Raekon. Anyone who use this dataset can adopt our labeled data or our data curation process to get the same data. **Commercial use is strictly forbidden**, and we recommend you to subscribe to this author and download the corresponding map environments to obtain the same dataset as ours. Author homepage: <https://www.patreon.com/Raekon>.

### C.3 Format

The folder structure of the *MineAnyBuild.zip* file we provide on Hugging Face (<https://huggingface.co/datasets/SaDil/MineAnyBuild>) is organized as follow:





```

├─ Task_Spatial_Reasoning.json
├─ Task_Spatial_Understanding.json

```

## C.4 Data Field Descriptions

Our dataset mainly consists of two major parts: architectures and tasks. “Architectures” refer to the buildings/decoration assets collected from the above-mentioned data resources, while “tasks” refer to five types of tasks that we designed for these “Architectures”. Each row in architectures and tasks data is organized as a structured JSON format. We provide the specific descriptions for each data field utilized in our dataset as follow.

### C.4.1 Architectures

The data field descriptions of architectures data are separated into two parts: GLOBAL for field common to all data and OPTIONAL for field only available in partial data.

GLOBAL:

- **id**: A unique identifier for an architecture. This identifier follows the format of {AR\_TDDDD\_hash1\_hash2}, where T indicates the types of data resources, and DDDD indicates the unique number of the architecture. *hash1* and *hash2* represent the first 16 hexadecimal digits of the hash values generated by the SHA-256 algorithm for information such as *types*, *name* and *description*. These hash strings are utilized as unique identifiers for an architecture/asset. (e.g.: AR\_00001\_4025eb7e4ff6ef93\_a9eab824ea6e85d9)
- **name**: A simple name of this architecture. For decoration assets, we set it as the format of “decoration\_asset\_{num}”.
- **description**: A textual description for the architecture or asset.
- **data\_resource**: An identifier selected from “Grabcraft”, “Minecraft Official Wiki” and “Raekon”.
- **3d\_info**: A dictionary that describes the 3D data of a cubic bounding box enclosing an entire 3D architectures/assets, e.g., {“width”: 5, “height”: 5, “depth”: 5}.
- **difficulty\_factor**: A difficulty coefficient obtained through comprehensive calculation based on the data distribution (introduced in Section D.3.2).
- **image**: A relative file address of a visualization image of the architecture/decoration asset.
- **block\_materials**: A list of all types of blocks utilized in the architecture/decoration asset. The list will be constructed into a mapping table (dictionary) during the construction process, using to associate **block\_materials** with the **blueprint** matrix.
- **blueprint**: A 3-dimension list (matrix) representing architectural construction blueprint through integers. Each integer corresponds to the index number of the **block\_materials** list plus 1. (e.g., for the list [A, B, C], the integer 2 corresponds to the block B.)

OPTIONAL:

- **type**: A field used to describe architecture types, which is not utilized to describe the indoor decoration assets. (e.g., the architecture “simple\_tree\_house” belongs to the type of “houses”)
- **biome**: A field exclusive to “Minecraft Official Wiki” data. Since these data come from the official resources, the Minecraft game has different construction plans and block types for similar building in different biomes (i.e., Minecraft natural environments).
- **image\_urls**: A field exclusive to “Grabcraft”, presenting the image urls that can be downloaded by http/https requests.
- **metadata**: A field to describe the data sources, which can be data urls or architectural metadata.

### C.4.2 Tasks

The data field descriptions of tasks data are separated into two parts: GLOBAL for field common to all data and OPTIONAL for field only available in partial data.

GLOBAL:

- **id**: A unique identifier for a task. This identifier follows the format of {TSK\_tasktype\_uniquestring}. *tasktype* is selected from “CR”, “SC”, “SP”, “SR” and “SU”, representing “Creativity”, “Spatial Commonsense”, “Executable Spatial Plan Generation”, “Spatial Reasoning” and “Spatial Understanding”, respectively. *uniquestring* is the unique string of different tasks. (e.g.: TSK\_SR\_1\_0084)
- **instruction**: An instruction for building architectures/assets or a question for Spatial Reasoning task and Spatial Commonsense task.

OPTIONAL:

- **AR\_id**: An identifier that is the same as **AR\_id** in Architecture data. This field is only set for Executable Spatial Plan Generation, Creativity and Spatial Understanding tasks.
- **difficulty\_factor**: A float value that is the same as **difficulty\_factor** in Architecture data. This field is only set for Executable Spatial Plan Generation, Creativity and Spatial Understanding tasks.
- **block\_materials**: A list that is the same as **block\_materials** in Architecture data. This field is only set for Executable Spatial Plan Generation, Creativity and Spatial Understanding tasks.
- **image**: One or more images that visualize the architecture/asset or perspective views. This field is only set for Executable Spatial Plan Generation, Spatial Understanding and Spatial Commonsense tasks.
- **image\_desp**: One or more textual descriptions of the **image**. This field is only set for Spatial Commonsense task.
- **options\_image**: An image containing a combination of 4 option images or 1 compared image, and an original image. This field is only set for Spatial Reasoning task.
- **options**: A list consisting of 4 option images. This field is only set for Spatial Reasoning task.
- **metadata**: A string that describes the option and selected stimuli. This field is only set for Spatial Reasoning task.

## C.5 Data Examples

In this subsection, we provide more visualization examples in our datasets.

### C.5.1 Architectures

We provide two visualization examples of architecture data in Figure 3.

### C.5.2 Tasks

We provide more visualization examples for the Executable Spatial Plan Generation, Spatial Understanding, Creativity, Spatial Reasoning, and Spatial Commonsense tasks, which are presented in Figure 4-8.

## C.6 Croissant Metadata

The Croissant file for our MineAnyBuild hosted on Hugging Face is available at <https://huggingface.co/api/datasets/SaDil/MineAnyBuild/croissant>. The file has successfully passed the the croissant-checker<sup>3</sup>.

<sup>3</sup><https://huggingface.co/spaces/JoaquinVanschoren/croissant-checker>



Figure 3: Visualization examples of architecture data.

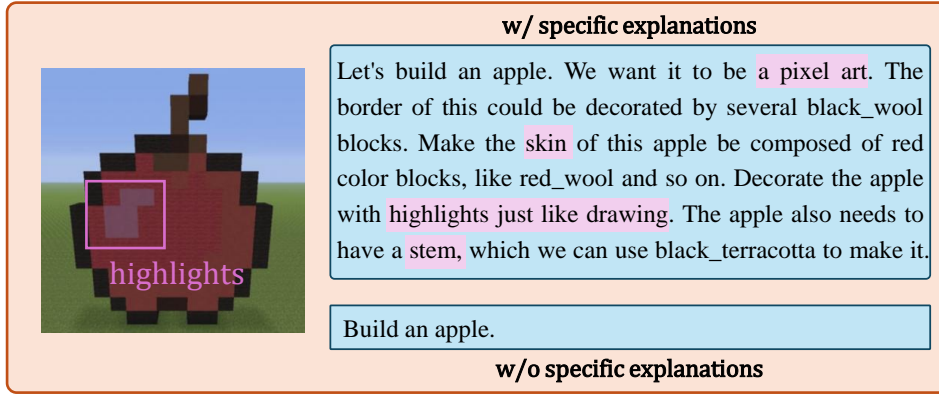


Figure 4: Task example visualization of Executable Spatial Plan Generation task.

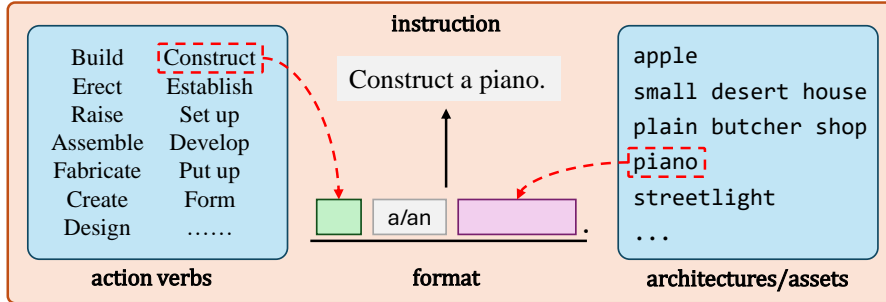


Figure 5: Task example visualization of Creativity task.

## D Details for Benchmark, Tasks and Data Curation

In this section, we provide some details about our benchmark MineAnyBuild, corresponding tasks and data curation pipeline.

### D.1 Details of Dataset

Our MineAnyBuild has 4000+ curated tasks covering several critical dimensions for evaluation of AI agents. We have curated 483 diverse architectures/assets from thousands of data candidates for planning tasks. We also collected 10+ large scenes for Spatial Commonsense task and generated 192 stimuli from the original datasets. The quantity and diversity of our datasets can also be infinitely expanded and scaled through our data curation pipeline. The numbers of different tasks in current

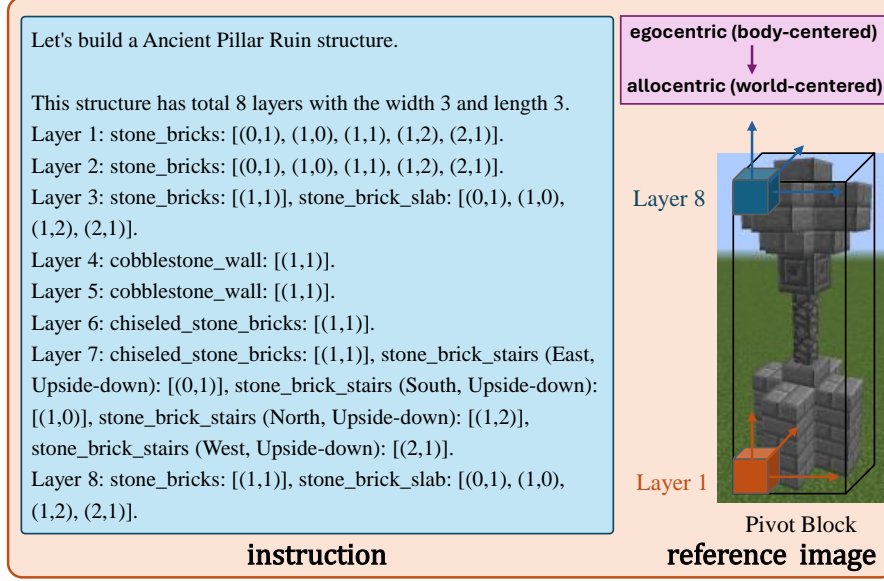


Figure 6: Task example visualization of Spatial Understanding task.

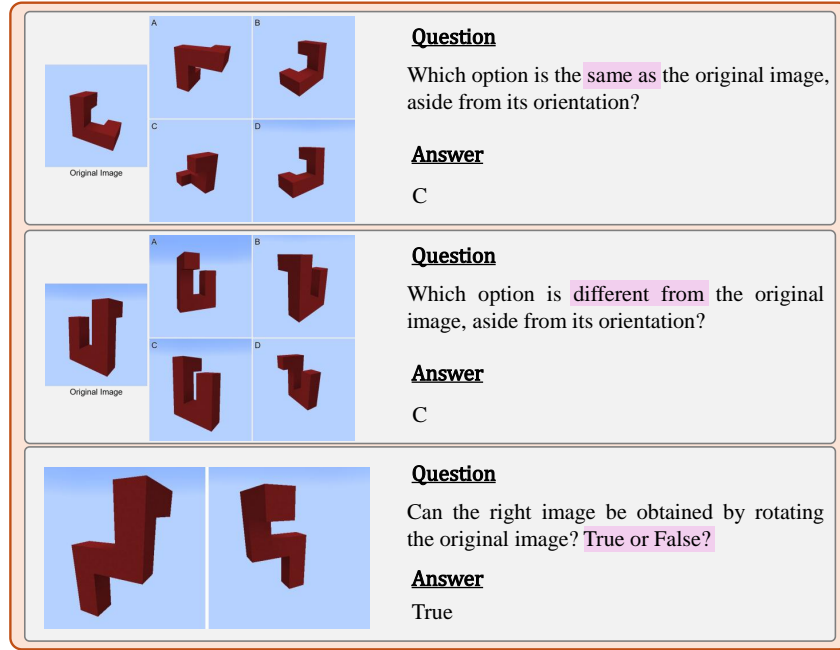


Figure 7: Task example visualization of Spatial Reasoning task.

MineAnyBuild datasets are shown in Table 3. There are 22 building types in our proposed dataset, including houses, fictional characters, items, etc., shown in Table 4.

Table 3: The numbers of different tasks in current MineAnyBuild dataset.

Task	Executable Spatial Plan Generation	Spatial Understanding	Spatial Reasoning	Creativity	Spatial Commonsense
Numbers	946	473	1728	1419	50

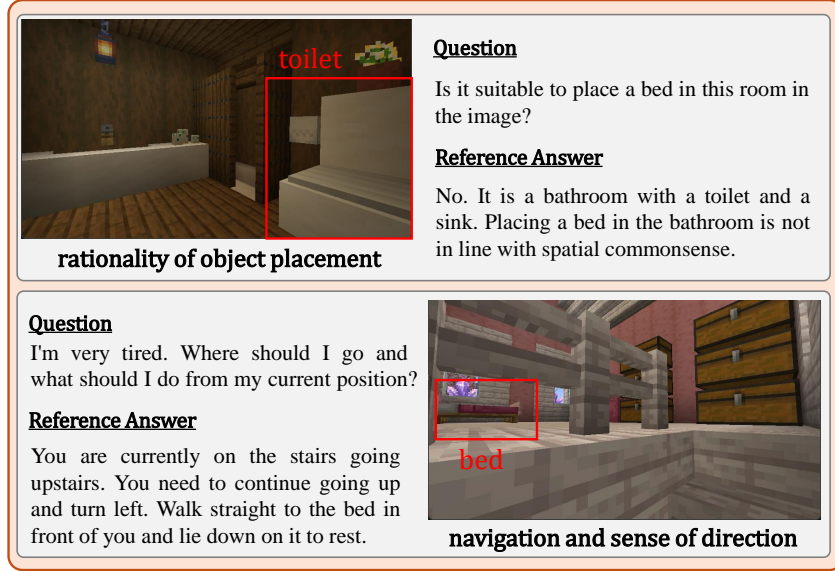


Figure 8: Task example visualization of Spatial Commonsense task.

Table 4: Building types in current MineAnyBuild datasets.

Types	houses, animals, items, town_centers, fictional_characters, cars, military_buildings, buses, emergency_vehicles, miscellaneous, famous_films, working_vehicles, bridges, garden, parks, ruins, ships, sightseeing_buildings, planes, minecraft_villages, lamps, others
-------	--

## D.2 Details of Tasks

### D.2.1 More Details of Executable Spatial Plan Generation Task

As we mentioned in the main text, the task input we designed consists of an abstract architecture building instruction accompanied by precise explanations. Agents are required to think about the decomposition of architectural substructures and their corresponding connections to generate executable spatial plans for architecture building, just like completing a jigsaw puzzle.

We also provide Executable Spatial Plan Generation tasks with simpler instructions. The instructions for this task type are consistent with those in Creativity task, only describing the entire building. We require the agent to output its planning based on the visual input and these simple instructions, and then provide the corresponding executable blueprint matrix according to the planning.

For tasks with given detailed instructions, we provide specific explanations based on the supervision by powerful MLLMs or human, guiding the agent to gradually construct these sub-structures according to the step-by-step explanations. While for simple instruction tasks without these detailed guidance, we require agents to output the process of decomposing substructures. In fact, these two types of tasks complement each other rather than being completely overlapping. Overall, they can effectively evaluate the agent’s comprehensive capability for spatial planning. We also provide examples of these two types of Executable Spatial Plan Generation task data in Figure 4 for differentiation.

### D.2.2 More Details of Creativity Task

As we mentioned in the Creativity task prompt in Section E.1, we guide MLLM-based agents on how to construct creative buildings. Specifically, the key points are as follows:

- 1) How to best interpret and implement the build specification.
  - List key elements from the build specification.
  - Brainstorm block combinations for different parts of the structure.
  - Outline a rough structure layout.
- 2) Creative ways to use the available blocks to achieve the desired aesthetic.
- 3) How to ensure the mapping correctness in your blueprint matrix.

- 4) Ways to maximize creativity and the dynamic range of possible builds.
- 5) Consider potential challenges and solutions.

In this way, we can guide agents to plan in a more creative direction, thereby stimulating their creative capabilities with specific prompts and presenting more distinctive architectural creations.

### D.2.3 More Details of Spatial Understanding Task

The instructions for the Spatial Understanding task can be automatically generated through templated code, i.e., we only need to associate the values of the matrix with the corresponding indices based on the existing ground-truth blueprint matrix and block materials, and set the block at  $[0][0][0]$  as the pivot block. Then we can obtain the corresponding task instruction data represented by relative positions.

It seems that this task is obvious and easy for the agent, but in fact it is not. During our conducted experiments, we found that the agents’ understanding of 3D data is still weak, and they have difficulties in handling the conversion between relative coordinates and world coordinates in practical applications. Therefore, we still keep this task to evaluate the agent’s perspective transformation capability, as shown in Figure 6.

The main difference between this task and the Executable Spatial Plan Generation task is that the former focuses more on the “result” of coordinate perspective transformation, while the latter emphasizes more on the “process” of spatial planning. The two tasks are not exactly the same but ultimately complementary.

### D.2.4 More Details of Spatial Reasoning Task

As shown in Figure 7, there are three main types of our Spatial Reasoning tasks in total, where selecting the same one from four stimuli, selecting the different one from four stimuli, and judging whether two stimuli can be obtained only by rotation.

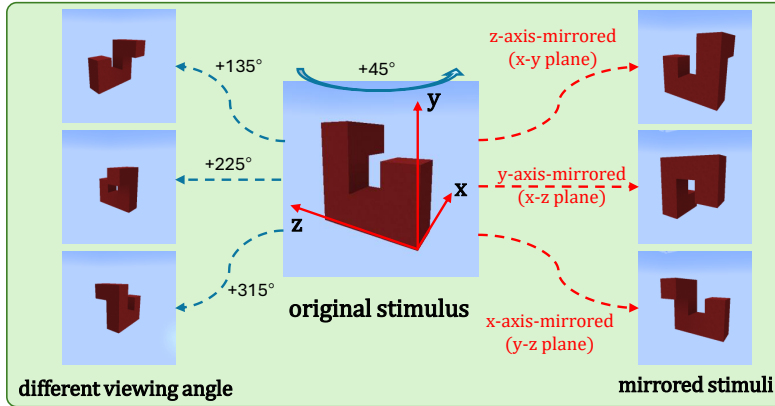


Figure 9: Augmentation of stimuli utilized in Spatial Reasoning task.

In Figure 9, we demonstrate how we expand from one stimulus to several stimuli. We mainly focus on randomly generated stimuli in the Minecraft environment and perform mirror symmetry along the X/Y/Z axes. The chiral geometries produced in this way usually cannot be made identical to the original ones just by rotation, unless the original stimulus itself has a certain degree of symmetry, but we require and avoid such geometries with obvious symmetry.

### D.2.5 More Details of Spatial Commonsense Task

Spatial Commonsense refers to humans’ intuitive understanding of spatial attributes such as the position, direction, distance, and shape of objects in the physical world, and it appears in every aspect of daily life. We list the most important types of spatial commonsense in our task data in Figure 10, and provide explanations as follows:

- **Rationality of object placement:** To estimate a contradiction between asset function and space and whether the size of an item is suitable for a certain space (e.g., “a refrigerator should not be placed in the bathroom”).
- **Navigation and sense of direction:** We require agents to locate directions without previous information, and determine abnormal situations (e.g., “you need to turn right when going upstairs” and “the chair can not be under the table”).
- **Path planning:** We require agents to avoid obstacles and bypass them rather than attempting to pass through a wall when conducting path planning.
- **Proximity commonsense:** We require agents to make judgments based on spatial commonsense and intuition rather than precise measurement, just as humans judge distance.
- **Furniture placement:** For some common combinations of spatial object relationships, such as the orientation and position of table and chair arrangements, we request agents to make correct judgments.
- **Spatial inclusion relationship:** If object A is completely inside of container B, then the volume of A must be smaller than the internal space of B (e.g., “a piano cannot fit into a chest”).
- **Transitivity of topological relations:** If A is to the left of B and B is to the left of C, then A is usually to the left of C. However, agents need to make judgments and verifications based on actual visual observations. Humans can determine this transitivity without logical training, and we evaluate whether the agents can achieve it.

Commonsense	Question
rationality of object placement	Is it suitable to place a bed in this room in the image?
navigation and sense of direction	I'm very tired. Where should I go and what should I do from my current position?
path planning	I'm walking now. What am I most likely to do?
proximity commonsense	Which is easier for me to reach, the bread in front or the cake in front?
furniture placement	Is the placement of the table and chairs in the current perspective appropriate?
spatial inclusion relationship	Can I carry this decoration asset that cannot be disassembled upstairs?
transitivity of topological relations	Based on my current perspective, the TV in front is to the left of the potted plant, and the potted plant is to the left of the pink cabinet. On which side of the TV is the pink cabinet?
:	:

Figure 10: Vital types of spatial commonsense in our task data.

### D.3 Details of Data Curation Pipeline

#### D.3.1 Details About Infinitely Expandable Paradigm

As we mentioned in main text, we conduct our data curation pipeline based on three steps: 1) data collection, 2) quality checking, and 3) data annotation. We also propose an infinitely expandable paradigm for our data curation pipeline.

Specifically, we manually mark the starting block (the minimum values on the X/Y/Z coordinates) and the ending block (the maximum values on the X/Y/Z coordinates) of the 3D coordinates as the three-dimensional coordinate box of the entire building, and obtain all the block information

corresponding to each position through *mineflayer* simulator [15]. After filtering the “air” blocks, corresponding *three\_d\_info*, *blueprint* and *block\_materials* can be acquired to obtain the architecture data. We visualize a simplified version of this pipeline in Figure 11.

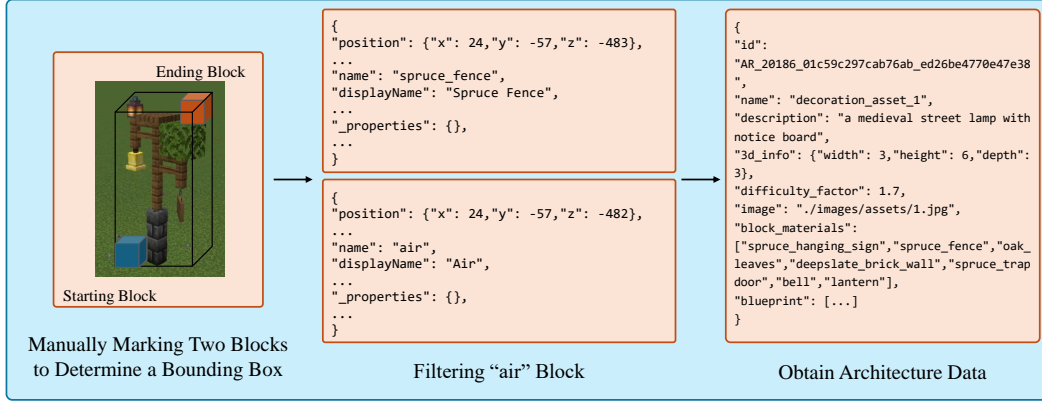


Figure 11: A simplified example of data curation pipeline.

### D.3.2 Estimation Metric of Difficulty Factor

To measure the construction difficulty for user-generated structures in Minecraft environment, we propose a novel estimation metric. This metric is designed to combine the perceived effort with complexity in the building process by integrating key architectural parameters. Its formulation is based on the following key considerations:

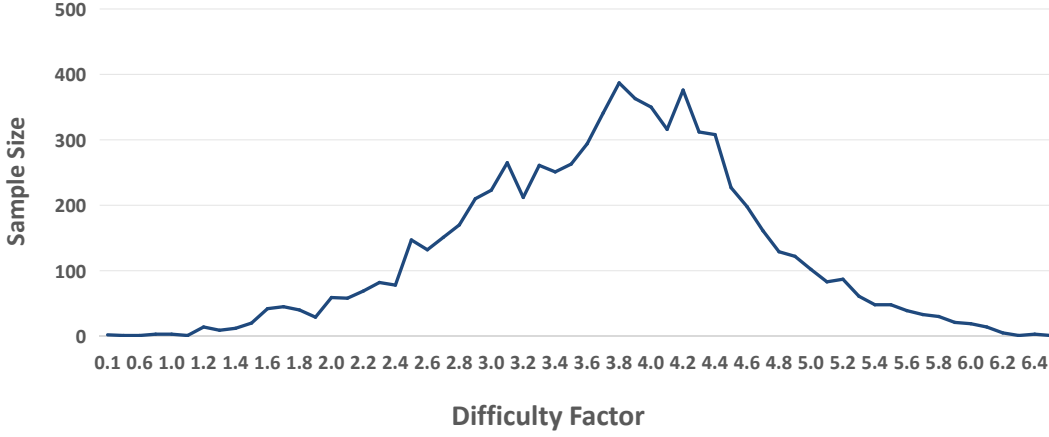


Figure 12: Difficulty Factor Distribution of task samples.

- **Base Workload:** The foundational construction effort is primarily determined by the total number of blocks ( $N$ ) required for the structure. A larger  $N$  typically corresponds to increased labor.
- **Architectural Scale and Volume Complexity:** The spatial extent of a structure, defined by its bounding box dimensions of length ( $L$ ), width ( $W$ ), and height ( $H$ ), significantly influences overall complexity. Larger volumes ( $L \cdot W \cdot H$ ) typically require more intricate planning, and attention to balance symmetry and structural integrity.
- **Height Challenges:** Building upward has extra difficulties compared to building outward. Considering in the same way as the commonsense of real-world architecture, factors such as the need for scaffolding, higher risk of falls, and the effort of moving materials vertically. This is captured with an interaction between the total block count ( $N$ ) and the height ( $H$ ), since placing blocks at greater heights is naturally harder.



Based on these factors, we define the difficulty estimation metric,  $D$ , as follows:

$$D = \ln(k_1 N + k_2 N H + k_3 L W H) - B$$

Where:

- $D$  is the estimated difficulty score, a relative scalar value where higher scores suggest greater theoretical construction complexity.
- $N$  represents the total count of blocks used in the construction.
- $L$ ,  $W$ , and  $H$  denote the values for length, width, and height of the structure’s bounding box, respectively.
- $k_1$ ,  $k_2$ , and  $k_3$  are weighting coefficients associated with the base workload, height-augmented workload, and volumetric complexity, respectively.
- $B$  is a bias term used for calibration.

In our initial tests, the coefficients  $k_1$ ,  $k_2$ , and  $k_3$  were all set to 1 (i.e.,  $k_1 = k_2 = k_3 = 1$ ), and the bias term  $B$  was set to 0.4. Early evaluations on a dataset of Minecraft constructions indicate that the resulting difficulty scores approximate a Gaussian (bell-shaped) distribution. This suggests that the metric can provide a solid basis for initial classification and comparison of construction difficulty.

Furthermore, the use of a logarithmic function, rather than mapping difficulty scores to a predefined fixed range, provides robust scalability to the metric. This makes it especially useful for new datasets, as it allows consistent evaluation across different levels of construction complexity and avoids problems like score saturation or low sensitivity at extreme values.

### D.3.3 Difficulty Tiers

As shown in Figure 12, our difficulty factor distribution is an approximately normal (Gaussian) but right-skewed distribution. Based on the distribution, we define three difficulty tiers using the mean  $\mu = 2.8$  and standard deviation  $\sigma = 1.0$ . Tasks with difficulty\_factor  $\leq 1.8$  are considered **Easy**, those between 1.8 and 3.8 are **Medium**, and those  $\geq 3.8$  are **Hard**, following the rule of  $\mu \pm \sigma$ .

Here is a simple experiment that we sample 50 data points and evaluate a baseline model (GPT-4o) in Creativity task with these difficulty tiers. The model achieves a score of 4.35 on Easy tasks, 2.95 on Medium tasks and 0.10 on Hard tasks. The result indicates that for easy tasks, agent can achieve a good score, while for hard tasks (e.g., to build an extremely complex house), it cannot complete correct execution and obtain a very low score.

## D.4 Discussion on Evaluation and Output Format

### D.4.1 Discussion on Evaluation and Output Format

We ultimately select the output format of a 3D blueprint matrix as the data structure to represent architectures. Why didn’t we choose the mainstream output formats, such as executable code? We considered that code generation is actually quite complex for architectures building, and this representation method cannot express the metadata form of buildings at the underlying level.

Although code generation aligns well with the motivation of disassembling sub-structures during the building construction process, for some complex buildings, this representation method only repeats the reasoning of agents, thus limiting the planning effect of agents. For example, when constructing a complex house with a roof, the code generation output by agents may be more inclined to express the coordinate values of each position in the matrix (e.g.,  $matrix[1][2][3] = Block\_A$ ). For some complex structures, they are more likely to make fundamental errors in the relationships between 3D coordinate points.

We expect agents to analyze and plan building outputs in the form of “Layers”, i.e., a 3D building is reduced in dimension to several 2D plane data through layering. For these 2D plane data, they can construct the building based on the characteristics of model data and the matrix as the root. If we also use code generation to express levels, we can find that this is consistent with the form of a blueprint. The blueprint method can ensure that agents better understand the relationships between

3D coordinate points, and then combine spatial reasoning at the abstract level to better reflect the spatial intelligence of agents.

The following is the reason why we do not present Spatial Reasoning and Spatial Commonsense tasks in planning forms.

Our MineAnyBuild datasets mainly include three tasks, i.e., Executable Spatial Plan Generation, Spatial Understanding, and Creativity to comprehensively evaluate the agents’ spatial planning ability in the aspects like instruction following or abstract architecture understanding. Beyond these tasks, we additionally supplement the VQA tasks of Spatial Reasoning and Spatial Commonsense for the intention of assisting the evaluation for the agents’ capabilities in commonsense-based spatial reasoning, which significantly impacts its spatial planning accuracy. Defining these two tasks with the VQA form is convenient for capability evaluation, which is also demonstrated in other previous benchmarks.

## D.4.2 Details of Output Format

### (1) Output Format of Blueprint Matrix:

The blueprint matrix has three dimensions. The three dimensions of this matrix are height (y-axis), depth (z-axis) and width (x-axis), respectively. The size of this matrix is determined by the sizes of the three dimensions.

The elements of the blueprint matrix are integers, each of which corresponds to a block. We use integers instead of strings as a representation of a block, to avoid an explosion in the number of tokens caused by a large number of elements. Therefore, we adopt a format similar to sparse matrices and use integers to represent elements of the matrix.

### (2) Output Post-processing for Architecture Generation:

We use the list "block\_materials" to represent the types of blocks that will be used in the building task. When querying the MLLM-based agents, we convert this list into a hash (a dictionary in Python) to indicate the mapping relationship between block types and integers. The "air" block is set to -1 by default. The integers corresponding to the block types are calculated by adding 1 to the index of this block type in the list. For example, for the list ["grass", "oak\_wood"], the hash should be "air": -1, "grass": 1, "oak\_wood": 2. We also use the reversed hash during the execution process to establish a reverse mapping for further evaluation and visualization.

## E Prompts

We provide task prompts, evaluation prompts, and data curation prompts in Section E.1, Section E.2, and Section E.3, respectively.

### E.1 Task prompts

#### (1) Executable Spatial Plan Generation:

##### **## System Prompt:**

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and Python coding. Your goal is to produce a Minecraft architecture, considering aspects such as architecture structure, block variety, symmetry and asymmetry, overall aesthetics, and most importantly, adherence to the platonic ideal of the requested creation.

##### **## User Prompt:**

Build the architecture based on the instruction and reference image. The instruction divides the architecture in the reference image by structure and demands. Please analyze and plan how to build the corresponding sub-structures according to the divided structure and demands, and give the ONLY ONE OVERALL blueprint matrix of the total architecture. The blueprint is a three-dimension list, and the dimensions of the blueprint matrix are in the order of height(positive y), length(positive z) and width(positive x). Fill in "-1" into the blueprint to denote as "air". The

elements of the list matrix are integers filled according to the given mapping table.

Here is an example.

Block materials: {"oak\_planks": 1}

Instruction: build a 3\*3\*4 (width, length, height) wooden house. We want it to be a simple boxy house. The roof and floor should be solid and there is some space that player can go inside the house.

Output:

Planning: The floor and roof of this wooden house can be made of 3\*3 oak\_planks as a square. Make the house hollow with air in the layer 2 and 3 and leave the space for entrance towards west (negative x). The two-layer walls are also made of 7\*2=14 oak\_planks. I can build it layer by layer so that I can truly understand the spatial structure of this house. Then, here is the overall blueprint matrix of the whole architecture. I'm sure that the 3-dim list has correct template and fill in -1 as "air" to simulate no block here.

Blueprint:

```
““json [[1,1,1],[1,1,1],[1,1,1]],[[1,-1,1],[1,-1,1],[1,1,1]],[[1,-1,1],[1,-1,1],  
[1,1,1]],[[1,1,1],[1,1,1],[1,1,1]]””
```

IMPORTANT: You must only use blocks from the given block materials dictionary. Make full use of them.

IMPORTANT: You must output ONLY ONE BLUEPRINT following the example format (A 3-DIM MATRIX).

IMPORTANT: Fill in -1(interger) as "air" into the blueprint matrix if no block is placed in the corresponding position.

### ## Input:

Now, take a breath and continue.

Block materials: {*block\_types*}

Instruction: {*instruction*}

Reference image: <image>

### ## Output:

Output:

## (2) Spatial Understanding:

### ## System Prompt:

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and Python coding. Your goal is to produce a Minecraft architecture, considering aspects such as architecture structure, block variety, symmetry and asymmetry, overall aesthetics, and most importantly, adherence to the platonic ideal of the requested creation.

### ## User Prompt:

Build the architecture based on the instruction and reference image. The instruction provides the relative coordinates of every block in reference image, and please identify the structure and summarize it as an overall blueprint matrix. The blueprint is a three-dimension list, and the dimensions of the blueprint matrix are in the order of height(positive y), length(positive z) and width(positive x). Fill in "-1" into the blueprint to denote as "air". The elements of the list matrix are integers filled according to the given mapping table.

Here is an example.

Block materials: {"oak\_planks": 1}

Instruction: build a 3\*3\*4 (width, length, height) wooden house layer by layer from bottom to top. Layer 1: oak\_planks: [(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)]. Layer 2: oak\_planks: [(0,0), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)]. Layer 3: oak\_planks: [(0,0), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)]. Layer 4: oak\_planks: [(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)].

Output:

```
'''json
[[[1,1,1],[1,1,1],[1,1,1]],[[1,-1,1],[1,-1,1],[1,1,1]],[[1,-1,1],[1,-1,1],[1,1,1]],[[1,1,1],[1,1,1],[1,1,1]]]]'''
```

IMPORTANT: You must only use blocks from the given block materials dictionary. Make full use of them.

IMPORTANT: You MUST output ONLY ONE BLUEPRINT following the example format (A 3-DIM MATRIX). You MUST NOT answer your reasons.

IMPORTANT: Your results should only follow the format of the example and not be influenced by the content of the examples.

IMPORTANT: Fill in -1(integer) as "air" into the blueprint matrix if no block is placed in the corresponding position.

#### ## Input:

Now, take a breath and continue.

Block materials: *{block\_types}*

Instruction: *{instruction}*

Reference image: <image>

#### ## Output:

Output:

### (3) Creativity:

#### ## System Prompt:

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and Python coding. Your goal is to produce a Minecraft architecture, considering aspects such as architecture structure, block variety, symmetry and asymmetry, overall aesthetics, and most importantly, adherence to the platonic ideal of the requested creation.

#### ## User Prompt:

Build the architecture based on the instruction and let your imagination run wild and use your creativity to build your best architecture. Before providing your final blueprint matrix, plan your solution considering the following spatial planning perspectives:

1. How to best interpret and implement the build specification.
  - List key elements from the build specification
  - Brainstorm block combinations for different parts of the structure
  - Outline a rough structure layout
2. Creative ways to use the available blocks to achieve the desired aesthetic.
3. How to ensure the mapping correctness in your blueprint matrix.
4. Ways to maximize creativity and the dynamic range of possible builds.
5. Consider potential challenges and solutions

The blueprint is a three-dimension list, and the dimensions of the blueprint matrix are in the order of height(positive y), length(positive z) and width(positive x). Fill in "-1" into the blueprint to denote as "air". The elements of the list matrix are integers filled according to the given mapping table.

Here is an example.

Block materials = ["oak\_planks", "cobblestone", "red\_wool"]

Instruction: build a 3\*3\*4 (width, length, height) wooden house.

Output:

Planning Reasons: Let's build a simple wooden house. I use cobblestone as the material for the floor and oak\_planks for the wall and roof.

Selected\_block\_materials = "oak\_planks": 1, "cobblestone": 2

Blueprint:

```
'''json
[[[2,2,2],[2,2,2],[2,2,2]],[[1,-1,1],[1,-1,1],[1,1,1]],[[1,-1,1],[1,-1,1],[1,1,1]],[[1,1,1],[1,1,1],[1,1,1]]]]'''
```

IMPORTANT: You must output ONLY ONE BLUEPRINT following the example format (A 3-DIM MATRIX).

IMPORTANT: Fill in -1(interger) as "air" into the blueprint matrix if no block is placed in the corresponding position.

**## Input:**

Now, take a breath and continue.

Block materials:  $\{block\_types\}$

Instruction:  $\{instruction\}$

**## Output:**

Output:

**(4) Spatial Reasoning:**

**## System Prompt:**

You are an expert Minecraft builder and player in a flat Minecraft Java 1.20.4 server.

**## User Prompt and Input:**

You need to answer this question with a visual image.

Question:  $\{instruction\}$  <image>

You must output ONLY one option (from True,False) without any reason based on the question.

IMPORTANT: You can only answer one word (from A,B,C,D or True,False).

**## Output:**

Your answer:

**(5) Spatial Commonsense:**

**## System Prompt:**

You are an expert in Minecraft and interior design, familiar with real-life common sense.

**## User Prompt and Input:**

You will receive a question and an image. Answer the question based on the image, focusing on spatial commonsense. Your response must not exceed 70 words. Do not include any additional content or thoughts. Now, take a breath and continue.

Instruction:  $\{instruction\}$

The next image is  $\{img\_desp\}$ . <image>

**## Output:**

Your answer:

**E.2 Evaluation Prompts**

**(1) Executable Spatial Plan Generation:**

**## System Prompt:**

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and an expert architecture critic.

**## User Prompt:**

Give a grade from 1 to 10 to the following Minecraft architectures from different views. The scores of reference human-annotated architectures are all 8 by default, as a reference for comparison. You should give the grade based on how well they are presented and correspond together to the building instructions in the following aspects:

- Completeness(Instruction Following): from \*nothing, abandoned\*(1) to \*partial, incomplete\*(5) and \*masterfully completed, perfectly realized\*(10).
- Complexity: from \*simplistic, basic\*(1) to \*straightforward, moderate \*(5) and \*challenging,

hardcore\*(10).

- Overall Aesthetic, Atmosphere and Fidelity: from \*stark, bare\*(1) to \*appealing, unusual\*(5) and \*epic, masterpiece\*(10).

### ## Input:

The following image is the ground-truth human-annotated reference image.

<image>

Give the grades based on the following image showing the Minecraft architecture in JSON format.

<image>

Building instructions: {*instruction*}

You must ONLY return the results in the following JSON format, but do not refer to the grades in this example and just follow the FORMAT:

```
{
  "Completeness(Instruction Following)": {
    "grade": 8,
    "comment": "The architecture well follows the building instructions and
    achieves a good finish, which is similar to the reference architecture."
  },
  "Complexity": {
    "grade": 8,
    "comment": "This architecture has several advanced building techniques like
    using several stairs upside down and half slabs to present some
    structures with half a block."
  },
  "Overall Aesthetic, Atmosphere and Fidelity": {
    "grade": 8,
    "comment": "The selection and placement of blocks have a certain aesthetic
    sense, reveal the feeling of ancient, in line with the requirements of
    the given instructions."
  }
}
```

You must not output any other reasoning or analysis.

### ## Output:

Output:

## (2) Spatial Understanding:

### ## System Prompt:

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and an expert architecture critic.

### ## User Prompt:

Give a grade from 1 to 10 to the following Minecraft architectures from different views. The scores of reference human-annotated architectures are all 10 by default, as a reference for comparison. You should give the grade based on how well they are presented and correspond together to the building instructions in the following aspects:

- Instruction Following(Completeness): from \*nothing, abandoned\*(1) to \*partial, incomplete\*(5) and \*masterfully completed, perfectly realized\*(10).

### ## Input:

The following image is the ground-truth human-annotated reference image.

<image>

Give the grades based on the following image showing the Minecraft architecture in JSON format.

<image>

Building instructions: {*instruction*}

You must ONLY return the results in the following JSON format, but do not refer to the grades in this example and just follow the FORMAT:

```
{
  "Completeness(Instruction Following)": {
    "grade": 8,
    "comment": "The architecture well follows the building instructions and
      achieves a good finish, which is similar to the reference architecture."
  }
}
```

You must not output any other reasoning or analysis.

### ## Output:

Output:

## (3) Creativity:

### ## System Prompt:

You are an expert Minecraft builder in a flat Minecraft Java 1.20.4 server and an expert architecture critic.

### ## User Prompt:

Give a grade from 1 to 10 to the following Minecraft architectures from different views. You should give the grade based on how well they are presented and correspond together to the building instructions in the following aspects:

- Creativity: from *\*boring, dull\*(1)* to *\*mediocre, normal\*(5)* and *\*blue sky thinking, inspiring\*(10)*.
- Completeness: from *\*nothing, abandoned\*(1)* to *\*partial, incomplete\*(5)* and *\*masterfully completed, perfectly realized\*(10)*.
- Complexity: from *\*simplistic, basic\*(1)* to *\*straightforward, moderate \*(5)* and *\*challenging, hardcore\*(10)*.
- Architecture Structure: from *\*boxy, rudimentary\*(1)* to *\*intuitive, modest\*(5)* and *\*sophisticated, intricate\*(10)*.
- Overall Aesthetic, Atmosphere and Fidelity: from *\*stark, bare\*(1)* to *\*appealing, unusual\*(5)* and *\*epic, masterpiece\*(10)*.

### ## Input:

Give the grades based on the following image showing the Minecraft architecture in JSON format.

<image>

Building instructions: *{instruction}*

You must ONLY return the results in the following JSON format, but do not refer to the grades in this example and just follow the FORMAT:

```
{
  "Creativity": {
    "grade": 6,
    "comment": "The building uses the same material but different forms of block
      types to enrich the architectural design. The design of this building is
      based on reality but beyond reality."
  },
  "Completeness": {
    "grade": 5,
    "comment": "The architecture well follows the building instructions and
      achieves a good finish. The architecture is not broken and is built
      completely."
  },
  "Complexity": {
    "grade": 6,
    "comment": "This architecture has several advanced building techniques like
      using several stairs upside down and half slabs to present some
      structures with half a block."
  },
  "Architecture Structure": {
```

```

    "grade": 6,
    "comment": "The design of the whole building is based on the vertical line as
                the axis and symmetrical in the center, which has the sense of extending
                upward."
  },
  "Overall Aesthetic, Atmosphere and Fidelity": {
    "grade": 5,
    "comment": "The selection and placement of blocks have a certain aesthetic
                sense, reveal the feeling of ancient, in line with the requirements of
                the given instructions."
  }
}

```

You must not output any other reasoning or analysis.

### ## Output:

Output:

## (4) Spatial Commonsense:

### ## System Prompt:

You are an expert in the field of multi-modal large language models and answer proofreading. You can well compare the differences between the output results of large models and the standard answers and score them.

### ## User Prompt:

You will get the output result of a multi-modal large language model and a standard answer. You need to compare the two and score the output result of the MLLM. What you need to note is:

1) Evaluate the matching degree between the output result of the large model and the standard answer. It is not necessary for the contents of the two to be completely the same, but the tendency of the answers must be the same to be considered a correct match.

2) You need to score the matching degree, with a full score of 10. If it is a correct match, please score at least 8 points or more. If it is a wrong match, please score at least 3 points or less. For example, if the output of the large model is yes, and the standard answer is no, then it is obviously a wrong match.

3) You need to carefully check the key information in the standard answer, such as spatial position and direction, action tendency, and spatial common sense reasoning. If the output of the large model meets all of them, please add points; if there are any that are not met, please deduct points as appropriate within the range.

Please output the score and scoring reason (within 70 words) following this JSON format:

"score": 5, "reason": ""

### ## Input:

Standard answer: {*answer*}

MLLM response: {*response*}

### ## Output:

Your result in JSON format:

## E.3 Data Curation Prompts

### ## System Prompt:

You are an expert in Minecraft architecture, proficient in spatial planning and architectural design.

### ## User Prompt:

Ensure that generated content is related solely to construction, excluding any irrelevant content related to user interaction.

Describe how to build this object in under 150 words.



Based on the input JSON file, generate a textual description of the structure within the file. The output must exclude any coordinates and adhere to natural language norms without structured bullet points or numbered lists.

Examples in the required format:

*Let's build a Ancient Pillar Ruin structure from bottom to top. First, place 5 stone\_bricks as the shape of a cross twice to build a two-layer foundation. Then, place 1 stone\_bricks in the center, and place each stone\_brick\_slab on the last 4 stone\_bricks. Place 2 cobblestone\_wall vertically reaching a height of two blocks, and then 2 chiseled\_stone\_bricks similarly on top of the cobblestone\_wall. Leaning against the top chiseled\_stone\_bricks, place 4 stone\_brick\_stairs upside down towards different orientation. Finally, place 1 stone\_bricks in the center, and place each stone\_brick\_slab on the 4 stone\_brick\_stairs.*

**## Input and Output:**

## F Additional Experimental Information, Results and Analyses

In this section, we first provide additional experimental information such as compute resources and evaluation metrics. Then, we present more quantitative results with corresponding analyses.

### F.1 Agents

Besides MLLM-based agents, reinforcement learning (RL)-based agents are also the mainstream research objects in many current related works on Minecraft environments. RL-based agents have a variety of low-level atomic actions and can be further developed in combination with reinforcement learning algorithms. We are planning to develop RL-based agents for addressing our proposed benchmark under popular RL frameworks such as Mineflayer [15], MineDojo [4], and MineRL [13]. We will synchronize and update these RL-based agents to our Github code repository at <https://github.com/MineAnyBuild/MineAnyBuild> in the future.

### F.2 Compute Resources and Cost

We mainly use MLLM-based agents for the evaluation of our benchmark MineAnyBuild.

For proprietary models, we implement the input/output (I/O) of the open-world AI agents by querying the corresponding APIs of MLLMs. We spent approximately 350 US dollars on the API calls in our current experiments.

For open-source models, we conduct the inference of these models on NVIDIA RTX 3090 GPUs and NVIDIA RTX 4090 GPUs. Noticing that the average output length of the planning tasks is much longer than that of other tasks, more GPU memories are required to support our benchmark evaluation. Considering the limited resources, we did not use GPUs with more memory to run open-source models with larger parameters. In the future, we will supplement the evaluation results of large-scale open-source models based on the actual situation.

### F.3 Evaluation Metrics

#### F.3.1 Scores

##### 1) Evaluation Scores:

During our experiments, we find that if we grade a single task from only one dimension, the scores of the critic model are not entirely reliable, making it hard to distinguish the fundamental differences among some outputs. Therefore, we consider introducing multiple scoring dimensions for evaluation. Inspired by Chain-of-Thought [24], the multi-dimensional scoring can prompt the critic model to provide more accurate scores.

We present the composition of evaluation scores and the corresponding scoring criteria (from 1 to 10) in the evaluation prompts in Section E.2. The evaluation scores are separated and designed to evaluate for each task.

**(a) Creativity.** We mainly have the following five scoring dimensions for evaluation: Creativity ( $S_{\text{Creativity}}$ ), Completeness ( $S_{\text{Completeness}}$ ), Complexity ( $S_{\text{Complexity}}$ ), Architecture Structure ( $S_{\text{AS}}$ ), Overall Aesthetic, Atmosphere and Fidelity ( $S_{\text{OAAF}}$ ).

$$S_{\text{Evaluation}} = k_1 * S_{\text{Creativity}} + k_2 * S_{\text{Completeness}} + k_3 * S_{\text{Complexity}} + k_4 * S_{\text{AS}} + k_5 * S_{\text{OAAF}} \quad (1)$$

where we set  $k_1 = 0.8, k_2 = k_3 = k_4 = k_5 = 0.05$  to better emphasize the evaluation of creativity.

**(b) Executable Spatial Plan Generation.** We mainly have the following three scoring dimensions for evaluation: Completeness(Instruction Following) ( $S_{\text{CIF}}$ ), Complexity ( $S_{\text{Complexity}}$ ), Overall Aesthetic, Atmosphere and Fidelity ( $S_{\text{OAAF}}$ ).

$$S_{\text{Evaluation}} = k_1 * S_{\text{CIF}} + k_2 * S_{\text{Complexity}} + k_3 * S_{\text{OAAF}} \quad (2)$$

where we set  $k_1 = 0.4, k_2 = k_3 = 0.3$  to balance various dimensions so as to comprehensively evaluate the effectiveness of spatial planning.

**(c) Spatial Understanding.** We only have the following scoring dimensions for evaluation: Completeness(Instruction Following) ( $S_{\text{CIF}}$ ).

$$S_{\text{Evaluation}} = S_{\text{CIF}} \quad (3)$$

For the evaluations scores above, we assign different importances to each dimension of a specific task based on the following principle: We first select a main scoring dimension based on human preference priors and assign it a relatively high importance. For the remaining dimensions, we evenly distribute the remaining weight proportion to ensure the output buildings have basic structures without obvious errors. The assignment of importance can achieve a relatively good balance between the main evaluation dimension of creativity task and other basic scoring dimensions.

## 2) Matching Scores:

For the tasks which correspond to a specific ground-truth blueprint of the reference images, we calculate the “Matching Score” between the results and ground-truths. We only conduct it on the Spatial Understanding and Executable Spatial Plan Generation tasks. The specific calculation formula for this score is as follows:

$$S_{\text{Matching}} = \frac{M}{N} * 10 \quad (4)$$

where  $N$  is the block amount of the ground-truth data, and  $M$  is the amount of matching blocks in the corresponding 3D bounding space filtering out “air” block. Only when the block types match in the relative position can it be considered valid.

We set the weighting coefficient of this score to a relatively small value between 0.05 and 0.1 because we found that most of the outputs could not perfectly reproduce the buildings compared to the original reference images. We believe that the construction of architectures does not necessarily have to be completely identical to be truly good. Therefore, we have adjusted this weighted value downward.

## 3) Voting Scores:

For Creativity task, we design a voting ranking scoring algorithm as “Voting Score” referencing the Swiss-round algorithm [25]. The specific design of the algorithm is shown in Algorithm 1. We set the weighting coefficient of this score to a relatively small value  $\sim 0.05$ . We will ultimately convert the rankings from  $L$  in Algorithm 1 into  $S_{\text{Voting}}$  ranging from 3 to 8 as the final voting score.

$$S_{\text{Voting}} = 3 + 5 \times \frac{\text{score} - \min(\text{score})}{\max(\text{score}) - \min(\text{score})} \quad (5)$$

We will also decide whether to use this score based on the specific scale of the evaluation.

## F.3.2 Formula for Comprehensive Score

For different tasks, we obtain a comprehensive score based on these three kinds of scores, denoted as “Score” (out of 10) shown in Table I of the main text, through corresponding weighting to indicate the performance of agents in each task. The specific calculation formulas for each task are as follows:

---

**Algorithm 1:** Multi-round Agent Battle Ranking Algorithm

---

**Require:** (1) Agent Ids  $I = id_1, \dots, id_n$ : The set of agent identifiers to be ranked; (2) Total Rounds  $R$ : The total number of battle rounds.

**Ensure:** (1) Ranked List  $L$ : The final ranked list containing (id, score); (2) Compared Log  $H$ : The complete battle record.

```
1: for each  $id_i \in I$  do
2:   Create a tuple of agents  $a_i \leftarrow \{id : id_i, score : 0, compared : \emptyset\}$ 
3: end for
4: for round  $r \leftarrow 1$  to  $R$  do
5:   Generate battle combinations:
6:   Sort the agent set by  $(-score, id)$ , and then create an empty battle set  $P \leftarrow \emptyset$  and a used set  $U \leftarrow \emptyset$ 
7:   for  $i \leftarrow 1$  to  $n$  do
8:     if  $a_i.id \in U$  then
9:       continue
10:    end if
11:    for  $j \leftarrow i + 1$  to  $n$  do
12:      if  $a_j.id \in U$  then
13:        continue
14:      end if
15:      if  $a_j.id \notin a_i.compared$  then
16:         $P \leftarrow P \cup \{(a_i, a_j)\}$ 
17:         $U \leftarrow U \cup \{a_i.id, a_j.id\}$ 
18:        break
19:      end if
20:    end for
21:  end for
22:  if  $P = \emptyset$  then
23:    break
24:  end if
25:  for each pair  $(a_a, a_b) \in P$  do
26:    Get Critic Model's response  $choice \in \{1, 2\}$ , Determine the winner  $w$  and the loser  $l$ :
27:    if  $choice = 1$  then
28:       $w \leftarrow a_a, l \leftarrow a_b$ 
29:    else
30:       $w \leftarrow a_b, l \leftarrow a_a$ 
31:    end if
32:    Update scores:  $w.score \leftarrow w.score + 1, l.score \leftarrow l.score - 1$ 
33:    Record the battle:  $w.compared \leftarrow w.compared \cup \{l.id\},$ 
     $l.compared \leftarrow l.compared \cup \{w.id\}$ 
34:  end for
35: end for
36:  $L \leftarrow$  The agent set sorted by the compare function
37: return  $(L, H)$ 
```

---

$$S_{CR} = k_1 * S_{Evaluation} + k_2 * S_{Voting} \quad (6)$$

$$S_{SP} = k_3 * S_{Evaluation} + k_4 * S_{Matching} \quad (7)$$

$$S_{SU} = k_5 * S_{Evaluation} + k_6 * S_{Matching} \quad (8)$$

where  $S_{CR}$ ,  $S_{SP}$  and  $S_{SU}$  are the comprehensive scores of Creativity, Executable Spatial Plan Generation, Spatial Understanding tasks, respectively. We set  $k_2 = k_4 = k_6 = 0.05$  with relatively low weights, and  $k_1 = 1 - k_2, k_3 = 1 - k_4, k_5 = 1 - k_6$  to emphasize the evaluation score.

For most cases, we only consider Evaluation Scores as the Comprehensive Score to simplify the evaluation. We will also dynamically adjust the proportions of these scores in the future and determine an optimal weighting coefficient based on experiments and manual supervision.

### F.3.3 Explanations on “Overall” Score

The “Overall” Score in Table 1 of the main text is the average score of these five tasks, with a maximum value of 100. The calculation formula is:

$$S_{\text{Overall}} = \frac{(S_{\text{ESPG}} + S_{\text{SU}} + Acc_{\text{SR}}/10 + S_{\text{C}} + S_{\text{SC}})}{5} * 10 \quad (9)$$

where  $S_{\text{ESPG}}$ ,  $S_{\text{SU}}$ ,  $Acc_{\text{SR}}$ ,  $S_{\text{C}}$  and  $S_{\text{SC}}$  are the Score/Accuracy of Executable Spatial Plan Generation, Spatial Understanding, Spatial Reasoning, Creativity and Spatial Commonsense tasks.

### F.3.4 Error Bar Chart of Scoring by Critic Models

We provide an error bar chart for the scoring of critic models, as shown in Figure 13. Specifically, we set 15 data points for the Creativity tasks, with different tasks for each data point. We conduct 10 API calls, using GPT-4.1 as critic model, with consistent inputs for each data point to obtain different outputs. For each score of the 10 outputs, we calculate the average, maximum, and minimum values to determine the upper and lower bounds of the error for that score. We set the upper bound of the error as the maximum value and the lower bound of the error as the minimum value, and then the values of the data point correspond to the average scores.

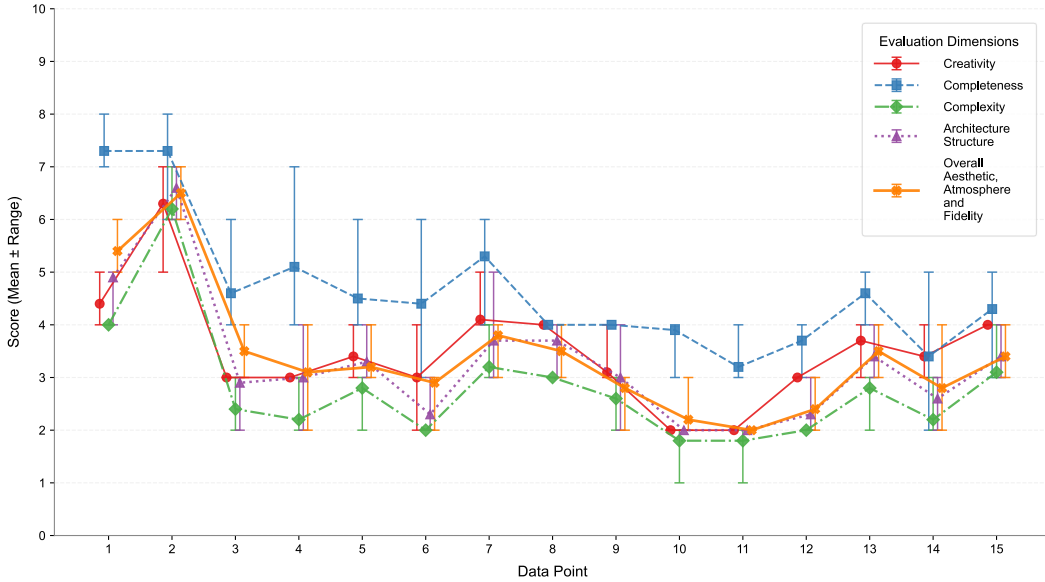


Figure 13: Error bar chart for the scoring of critic models.

We can observe that for most data points, the upper and lower error of the scores is concentrated within 1. The scores given by the critic model are relatively consistent, which also lays a foundation for the reliability of our benchmark.

## F.4 Assessments for Reliability Evaluation

We provide some assessments to present the reliability and reasonability of the critic model in our benchmark.

### (1) Spearman Correlation:

To evaluate the reliability of our MLLM-based critic model, we randomly sample 50 task outputs from our benchmark. We rate these outputs from 1 to 10 just like the critic model does. We compute the Spearman correlation between human and model scores, obtaining  $\rho = 0.76(p < 0.01)$ . The value shows that the reliability and reasonability of the critic model.

### (2) Intra-model Consistency:

As shown in Figure 13, we can observe that for most data points, the upper and lower error of the scores is concentrated within 1. The scores given by the critic model are relatively consistent, which also lays a foundation for the reliability of our benchmark.

## F.5 Full Results of Output Success Rate

We provide the detailed data corresponding to Figure 4 in the main text, as shown in Table 5.

Table 5: Detailed data of Output Success Rate (OSR) in Figure 4 of the main text.

Models	Spatial Understanding	Executable Spatial Plan Generation	Creativity
<b>Proprietary</b>			
Claude-3.5-Sonnet	97.41	98.91	100.00
Claude-3.7-Sonnet	94.59	96.36	98.70
Gemini-1.5-Flash	73.81	93.25	98.27
Gemini-1.5-Pro	94.48	99.56	93.66
Gemini-2.0-Flash	91.23	86.37	94.48
GPT-4o	88.09	92.44	92.03
GPT-4o-mini	95.73	95.10	86.94
<b>Open-source</b>			
InternVL2.5-2B	34.95	27.60	28.18
InternVL2.5-4B	37.90	32.71	42.18
InternVL2.5-8B	74.33	68.82	66.54
Qwen2.5VL-3B	50.88	50.84	44.65
Qwen2.5VL-7B	76.46	77.63	79.42
LLava-Onevision-7B	42.87	52.87	60.38
<b>Abandoned</b>			
InternVL2.5-1B	16.61	21.20	22.19
LLava-Onevision-0.5B	13.21	14.76	15.23

## F.6 Accuracy Evaluation Results on Spatial Reasoning Task

We present the results of more MLLM-based agents on Spatial Reasoning task, as shown in Table 6. We can observe that even some state-of-the-art (SOTA) MLLMs, e.g., GPT-4.1 and Gemini-2.5-Pro, do not perform optimally on our spatial reasoning tasks. Although the accuracy of these SOTA models on these tasks is not high, the accuracy on human evaluation (where task difficulty is simpler than ours) is only 60-90% [26], which also indicates the high difficulty of the task itself. This result also reveals that MLLM-based agents are required to be trained on tasks related to spatial intelligence to further improve their intelligence level in the future.

Table 6: Accuracy evaluation results of more MLLM-based agents on the Spatial Reasoning task.

Models	Accuracy ↑
<b>Proprietary</b>	
Gemini-2.5-Pro	24.02
GPT-4.1	20.25
GPT-4.1-mini	22.57
<b>Open-source</b>	
InternVL2.5-26B	26.16

## F.7 Discussion on Human Evaluation

We conducted a simple experiment to examine the differences between human evaluation and the evaluation by critic models. We asked 5 volunteers to score 10 or 30 architectures (with 13 results for each architecture) via a questionnaire. We collect the following findings in the process of organizing the questionnaire results:

- 1) Human evaluation is prone to inertia and fatigue during multiple evaluations, which can affect

judgment. Especially when we changed the number of architectures to be evaluated from 10 to 30, most volunteers were fatigued by the questionnaire’s length, which can easily lead to evaluation bias.

Human evaluation is intuitively more convincing than critic models. However, considering the large-scale evaluation needs in the future, such a heavy workload is more suitable to be handled by the highly intelligent critic model.

2) Human evaluation usually requires a reference. People usually select one result from multiple similar ones as a reference and score others with minor variations based on this reference.

3) Humans are not very accurate in scoring, and each person has different standards. But when the quantity and scale are sufficient, human evaluation remains highly reliable.

## **F.8 Discussion on Possible Noise Brought by Critic Model**

For analysis of the possible noise, we find that there are mainly two probable aspects.

**(1) Different Scoring Comments:** We require the critic model to output the comments while giving the evaluation scores. As shown in the evaluation prompts in Section E.2 of Supplementary Material, we manually provide the output examples and the comments generated by critic model are possible sources of noise.

**(2) Manually Customized Scoring Tiers:** As shown in our evaluation prompts, we provided some words for ratings of 1, 5 and 10 respectively, as scoring references for the critic model. The manually-written prompts in this part may introduce some possible noise.

But overall, the impact of these two possible sources of noise on our evaluation is marginal, which can be supported by the provided error bar chart.

## **F.9 Discussion on Abnormal Result on GPT-4o-mini in Spatial Reasoning Task**

The evaluation results in Spatial Reasoning task of other models (e.g., GPT-4o) are normal, while the output results of GPT-4o-mini are abnormally uniform. Due to the smaller parameters and weaker capabilities of GPT-4o-mini, most of the outputs are the frequent answers (e.g., B/C/False with 765/640/288 times), and A/D/True (with 10/24/0 times) are rarely seen. It shows that GPT-4o-mini is more likely to “guess” a relatively higher accuracy in this test but the absolute score is not high, though our data is evenly and randomly distributed across all options. In contrast, stronger models may attempt to reason more deeply, which may not always get higher scores on difficult questions.

## **F.10 Information and Metrics about Quality Control Processes in Data Curation Pipeline**

We reanalyze our data generated before and provide the lacking information and metrics including rejection rates and inter-annotator agreement about quality control processes for Executable Spatial Plan Generation task as follows.

**(1) For rejection rates:** We randomly sample 100 instructions from this task and observe a rejection rate of 11%, suggesting that most machine-generated instructions are broadly acceptable with minimal invalid cases. In most of the rejected cases, information of the instructions is either too redundant or contains some extraneous information. Thanks to the powerful instruction generation ability of GPT-4.1, the quality of most generated instructions is already quite high. Our manual processing is just to ensure that they meet our expectations.

**(2) For inter-annotator agreement (IAA):** We randomly select 50 instructions from this task and ask two annotators to judge whether each instruction is acceptable/rejected. We calculate Cohen’s Kappa to evaluate IAA and obtain a score of  $\kappa = 0.63$ , indicating a strong level of agreement and the reasonably high quality of machine-generated task instructions.

## **F.11 Discussion on Other Baseline Evaluation**

Our MineAnyBuild mainly focuses on evaluating MLLM-based agents, which is highly representative for the evaluation of AI agents. The baselines including human experts, domain-specific and reinforcement learning-based agents are well worthy of being discussed and researched in the

future. Due to several engineering challenges of code implementation and tight time constraints, we provide some transferring insights below for future research:

**(1) For human experts:** We plan to recruit  $\sim 10$  human participants with experience in Minecraft building by referencing current work [27], to perform the same planning tasks to build the architectures. We will utilize the same evaluator and evaluation criteria to assess human experts and agents, so as to provide more comprehensive evaluation results.

**(2) For domain-specific agents:** Domain-specific agents, such as Voyager [7], could potentially perform well by leveraging task-specific knowledge or predefined strategies on Minecraft tech-tree tasks. The input/output of these agents cannot be directly transferred to our benchmark, therefore we need to do some adaptation works for further evaluation. Constructing some in-domain agents for our building data leaves some future works to better improve the intelligence of AI agents.

**(3) For reinforcement learning-based agents:** In Section F.1 and the GitHub repository, we have stated that we plan to develop RL-based agents under popular RL frameworks (e.g., Mineflayer [15], MineDojo [4], and MineRL [13]) in the future. These agents will adopt an architecture similar to Vision-Language-Action (VLA) models. These VLA models will focus on how to perform action modeling on vision and language inputs and embeddings, which will bring certain help to the research of AI agents other than language modeling. We believe that this research will also be helpful for the research in Embodied AI in the future.

## F.12 Performance of Small-parameter Open-source MLLMs

For some models with small parameters (e.g., 0.5B and 1B), we found that their performance was particularly poor and errors often occurred. Therefore, we finally decided not to include these results in the main text. For the sake of fairness, we placed these results in Table 7.

Table 7: Evaluation results of small-parameter open-source MLLMs on **MineAnyBuild**.

Models	Executable Spatial Plan Generation	Spatial Understanding	Spatial Reasoning	Creativity	Spatial Commonsense	Overall
	Score $\uparrow$	Score $\uparrow$	Accuracy $\uparrow$	Score $\uparrow$	Score $\uparrow$	
<i>Open-source</i>						
InternVL2.5-1B	0.11	0.08	28.5	0.12	4.18	14.68
LLava-Onevision-0.5B	0.09	0.00	27.8	0.37	3.26	13.00

Regarding the performance of these models on the Spatial Reasoning task, we find that in most cases, their outputs are the same option (e.g., “A” or “False”), and only in a few cases they output other options. This results in a probability for them to achieve an accuracy slightly higher than 25%, which is not worthy of reference when compared with other models.

## G Additional Visualization Results

In this section, we provide additional visualization results with analyses about the executable planning output and failure cases.

### G.1 Executable Planning Output

We provide more visualization results on executable planning output in Figure 14. We can see that proprietary models perform well in spatial planning tasks of complex architectures. For example, for *palm tree* and *Chinese arch* in Figure 14, both GPT-4o and Gemini-2.5-Flash can well convert relative coordinates to world coordinates in Spatial Understanding task. For Creativity task, the agents exhibit capabilities to manifest architectural structures without ground-truth visual supervision. As shown in the Figure 14 below, the agent proactively utilize the block materials to construct the desired architecture by spatial planning and unleashing creativity.

### G.2 Failure Cases

We provide more visualization results on failure cases in Figure 15, revealing that there are still some challenges for agents, e.g., spatial misunderstanding, chaotic/fragmented/disorganized generation, and homogeneous boxy structure output. We can observe that, when the block amount exceeds a certain

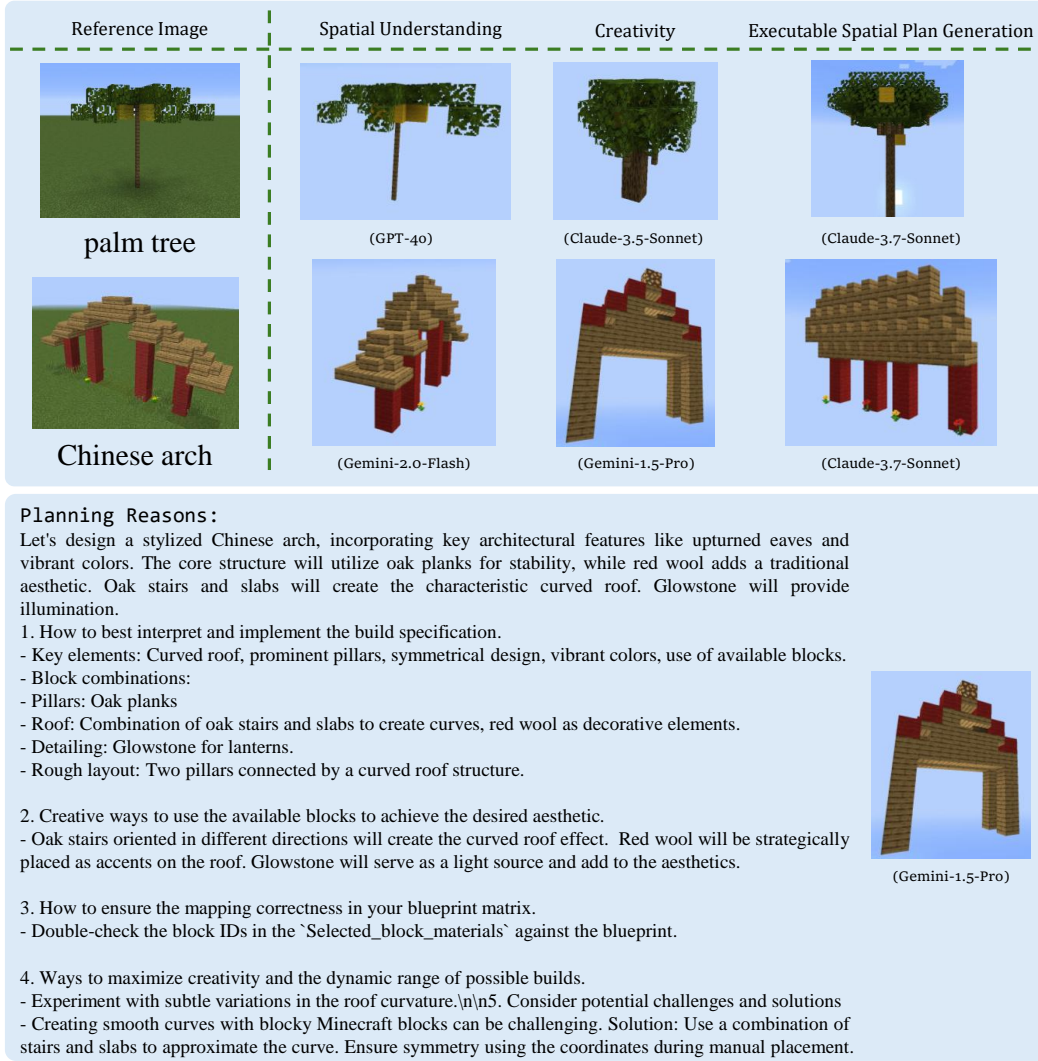


Figure 14: More visualization results on executable planning output.

threshold or facing non-cubic architectural structures, agents struggle to understand precise space localization and relationships among blocks, particularly in sub-structure composition and design. Furthermore, agents present limited capability to interpret relationships between block orientation (e.g., *stairs* as a part of the house roof) and global structural patterns. It is worth emphasizing that the fundamental issue lies in agents' abilities to "translate" textual planning into executable architectural outputs (i.e., blueprint matrix), namely implementation gap. These identified limitations provide future directions for spatial intelligence research.

Moreover, we provide the deep analyses of the failure reasons for the cases in the main text and above with a summarization as follows:

**(1) Spatial Misunderstanding:** Agents frequently misinterpret 3D positional relationships or fail to maintain the correct spatial arrangements, which highlights a persistent weakness in spatial grounding and planning.

**(2) Implementation Gap:** The agents have a central issue that they can not transform high-level textual plans into precise and executable blueprint matrices. The integration of substructures often fails due to incorrect block indexing, orientation errors or inconsistent spatial logic, leading to blueprint parsing or execution failures. This is essentially because the model's understanding of data structures such as codes or DSL and 3D matrices from a numerical perspective is still limited. If these models strengthen the training of spatial data, it may enhance the capabilities of these agents.



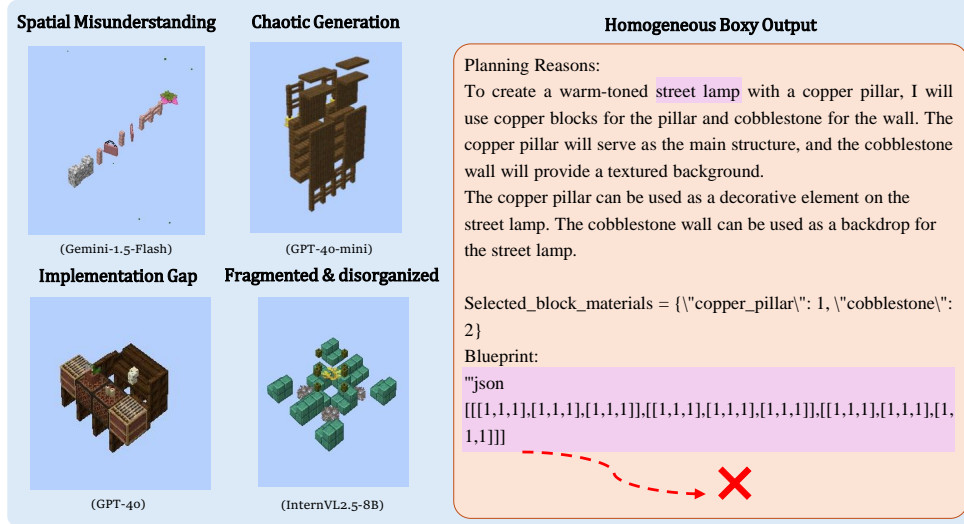


Figure 15: More visualization results on failure cases.

**(3) Structural Degeneration under Complexity:** When the tasks demand non-cubic, asymmetric or creative designs, the agents tend to collapse into simple and box-like outputs or disorganized results. This indicates that their limited ability to scale from basic patterns to more abstract and complex architectural concepts.

These failure modes reflect deeper limitations in MLLM’s capabilities to perform hierarchical spatial planning, maintain geometric consistency and ground language into manipulable 3D structures. They also provide more research directions for MLLMs, e.g., to improve multi-modal spatial understanding, align linguistic abstraction with executable plans or enhance agent’s ability for structural composition in open-ended 3D environments.

## H Declaration of LLM Usage

We primarily use LLMs for data filtering and the scoring system in our critic model (Multi-modal Large Language Models). Our implementation involves no ethical breaches, with all outputs conducting human review to ensure safety and compliance. The usage of LLMs in our work does not impact the scientific rigorousness and originality of the research.

## References

- [1] Paolo Faraboschi, Eitan Frachtenberg, Phil Laplante, Dejan Milojicic, and Roberto Saracco. Artificial general intelligence: Humanity’s downturn or unlimited prosperity. *Computer*, 56(10):93–101, 2023.
- [2] Kenneth M Heilman, Stephen E Nadeau, and David O Beversdorf. Creative innovation: possible brain mechanisms. *Neurocase*, 9(5):369–379, 2003.
- [3] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.
- [4] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.
- [5] Junliang Guo, Yang Ye, Tianyu He, Haoyu Wu, Yushu Jiang, Tim Pearce, and Jiang Bian. Mineworld: a real-time and open-source interactive world model on minecraft. *arXiv preprint arXiv:2504.08388*, 2025.
- [6] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.
- [7] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- [8] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *arXiv preprint arXiv:2408.03615*, 2024.
- [9] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy. *arXiv preprint arXiv:2502.19902*, 2025.
- [10] Shaofei Cai, Zhancun Mu, Anji Liu, and Yitao Liang. Rocket-2: Steering visuomotor policy via cross-view goal alignment. *arXiv preprint arXiv:2503.02505*, 2025.
- [11] Qian Long, Zhi Li, Ran Gong, Ying Nian Wu, Demetri Terzopoulos, and Xiaofeng Gao. Teamcraft: A benchmark for multi-modal multi-agent systems in minecraft. *arXiv preprint arXiv:2412.05255*, 2024.
- [12] Shaofei Cai, Zihao Wang, Kewei Lian, Zhancun Mu, Xiaojian Ma, Anji Liu, and Yitao Liang. Rocket-1: Mastering open-world interaction with visual-temporal context prompting. *arXiv preprint arXiv:2410.17856*, 2024.
- [13] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.
- [14] Shaofei Cai, Zhancun Mu, Kaichen He, Bowei Zhang, Xinyue Zheng, Anji Liu, and Yitao Liang. Minsstudio: A streamlined package for minecraft ai agent development. *arXiv preprint arXiv:2412.18293*, 2024.
- [15] PrismarineJS. Mineflayer. <https://github.com/PrismarineJS/mineflayer>.
- [16] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36:69900–69929, 2023.

- [17] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [18] Shaofei Cai, Bowei Zhang, Zihao Wang, Haowei Lin, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot-2: Weakly supervised multi-modal instruction following agents. *arXiv preprint arXiv:2412.10410*, 2024.
- [19] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.
- [20] Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16307–16316. IEEE, 2024.
- [21] Jun Yu Chen and Tao Gao. Apt: Architectural planning and text-to-blueprint construction using large language models for open-world agents. *arXiv preprint arXiv:2411.17255*, 2024.
- [22] GrabCraft LLC. Grabcraft - the biggest library of minecraft objects, models, floor plans, ideas, and blueprints. <https://www.grabcraft.com/>.
- [23] Fandom. Fandom wiki for minecraft. [https://minecraft.fandom.com/wiki/Minecraft\\_Wiki](https://minecraft.fandom.com/wiki/Minecraft_Wiki).
- [24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [25] László Csató. On the ranking of a swiss system chess team tournament. *Annals of Operations Research*, 254(1):17–36, 2017.
- [26] Nadia M Bersier, Eleonora Fornari, Raffaella I Rumiati, and Silvio Ionta. Cognitive traits shape the brain activity associated with mental rotation. *Cerebral Cortex*, 35(4):bhaf069, 2025.
- [27] Yunfan Jiang, Ruohan Zhang, Josiah Wong, Chen Wang, Yanjie Ze, Hang Yin, Cem Gokmen, Shuran Song, Jiajun Wu, and Li Fei-Fei. Behavior robot suite: Streamlining real-world whole-body manipulation for everyday household activities. *arXiv preprint arXiv:2503.05652*, 2025.