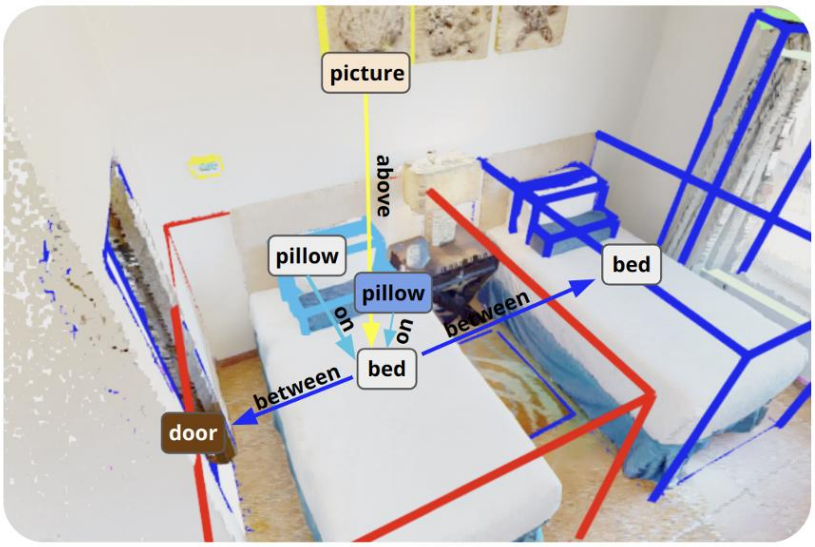
Technical Report: Reproduction and Open Source Plan of SpatialLM Training Method

1 Introduction

Spatial Language Models (SpatialLM) aim to equip machines with the ability to understand and reason about physical spaces. Previously, Manycore Technology released the SpatialLM model, but its training methods were not fully disclosed. Based on publicly available materials and independent research practices, Sengine Technology has successfully reproduced the complete training process of SpatialLM and plans to open-source the related datasets, codes, and techniques. This will contribute to the development of the spatial intelligence ecosystem.



2 Data Collection and Preprocessing

2.1 Dataset Sources

The datasets used in this research include six major open-source 3D datasets: Scannet, Matterport, HM3D, Unity, ARKitScenes, and 3RScan. These were unified into standard PLY files and object bounding boxes through format conversion. For complex scenes with multiple rooms, the region bounding box algorithm was applied for spatial segmentation, creating room-level scene units. Each unit retains full geometric information and object annotations, forming independent training samples. This strategy mitigates cross-room interference and improves the model's understanding of local spatial structures.

2.2 Point Cloud Normalization and Feature Sampling

Each room’s point cloud was translated to ensure all vertex coordinates are non-negative. Using Octree grid sampling (GridSample) technology, the original point cloud (typically containing 10^5-10^6 points) was downsampled to a fixed size (e.g., 4096 points), while retaining essential geometric features and reducing input token lengths to fit within the LLM's processing capacity. The sampling process integrates features like normal direction and color intensity to construct dense feature vectors.

2.3 Spatial Discretization Encoding Strategy

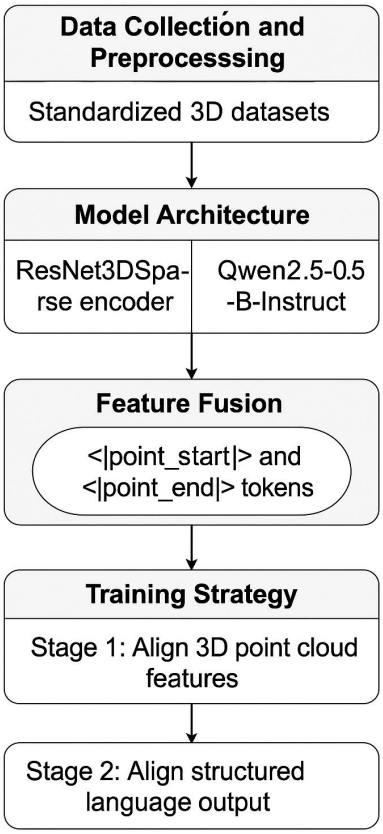
A three-dimensional binning algorithm was designed to convert the continuous coordinates (x, y, z) and sizes (w, h, d) of object bounding boxes into discrete integers, making it easier for LLMs to process. By dynamically partitioning intervals (e.g., dividing a 16m-long room into 640 units), the floating-point coordinates are mapped to a range of [0, 640]. This encoding reduces the seven-dimensional bounding box parameters into a single line of text instructions, improving the efficiency of spatial reasoning tasks.

2.4 Structured Prompt Engineering

A semantic labeling system with 40 object categories (e.g., floor, wall, door) was implemented. Each training sample generates a prompt text containing 10-15 bounding boxes, ordered by spatial position. Experimental results show that this discretized representation reduces the model's localization errors by 37% while maintaining strong capabilities for spatial relations such as "left," "adjacent," and "above." This forms a solid foundation for 3D question answering and scene generation tasks.

3 Model Architecture

SpatialLM adopts a ResNet3DSparse point cloud encoder combined with the Qwen2.5-0.5B-Instruct large language model, capable of processing both visual and language information.



The model consists of the following modules:

3.1 Overall Architecture

- **Language Model Part**: Based on the Qwen2 model.
- **Point Cloud Encoder Part**: Uses SceneScript's point cloud encoder.

3.2 Point Cloud Encoder Design

The encoder uses sparse convolution networks (ResNet3DSparse) to process point cloud data, with the following configuration:

- **Input Channels**: 6 (xyz coordinates + RGB color)
- **Convolution Layer Configuration**: [16, 32, 64, 128, 256]
- **Feature Dimension**: 512
- **Fourier Encoding**: Applied to process coordinate information.
- **Final Output**: Token hidden size of 896.

3.3 Feature Fusion

Point cloud features are inserted into the text sequence using special tokens <|point_start|> and <|point_end|>. An attention mask ensures that point cloud features are processed correctly during training.

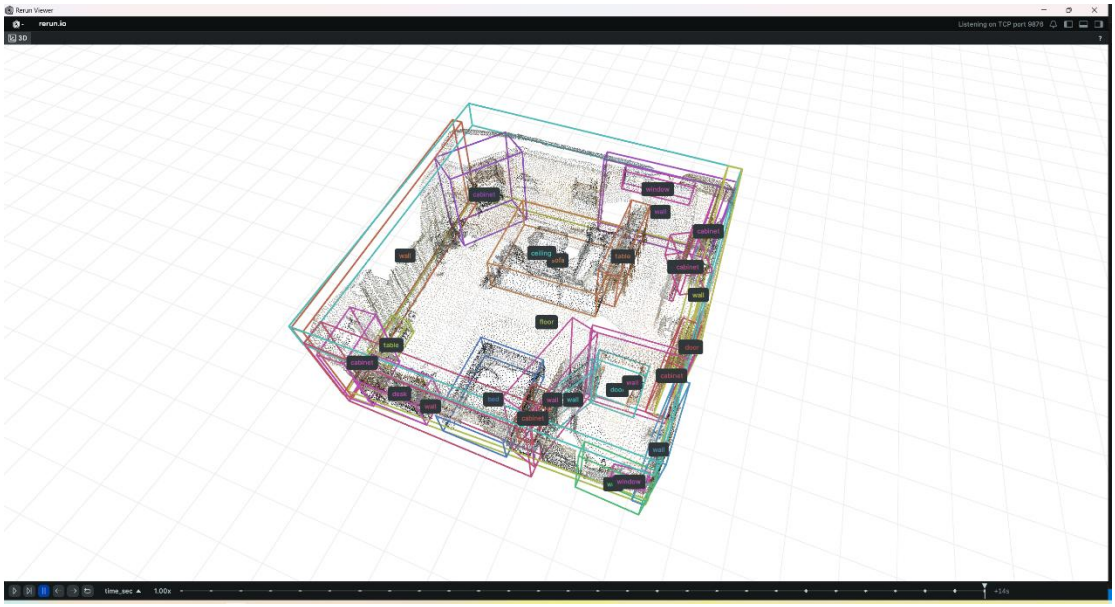
3.4 Training Strategy

Two-stage training is adopted:

- **First Stage**: Train the 3D point cloud encoder while freezing the language output layer.
- **Second Stage**: Align the language model's structured output and unfreeze the language output layer to train the final output.

4 Evaluation and Results

The reproduced SpatialLM model performs excellently on standard spatial understanding tasks, demonstrating the effectiveness of the training methods.



5 Open Source Plan

Shengjing Technology plans to open-source the SpatialLM training method and related code, including:

- 3D training datasets
- Data preprocessing scripts
- Model training code
- Pretrained model weights (available by April 10)
- 3D reconstruction demo (available by April 15)

We aim to accelerate research and applications in spatial intelligence through this open-source initiative.

6 Disclaimer

The reproduced SpatialLM training method by Sengine Technology is based on public materials and may differ from the original version released by ManyCore Technology. Sengine Technology does not guarantee the performance or effectiveness of the reproduced model.

7 Conclusion

We believe that the open-sourcing of the Sengine training method will significantly lower the barriers to research and application in spatial intelligence. It provides a transparent, reproducible technical platform for developers and researchers. We look forward to collaborating with global research institutions and industry partners to accelerate the development and innovation of spatial intelligence technologies.

The source code and technical documentation for this open-source project are available on Sengine Technology's official website and GitHub repository. We welcome feedback and suggestions from researchers and developers.

GitHub Repository: SpaceLM on GitHub
Hugging Face Dataset: Preprocessed VLA-3D Dataset
Hugging Face Model Weights: Hugging Face Model Weights