SpatialLM: Training Large Language Models for Structured Indoor Modeling

Yongsen Mao[1]* **Junhao Zhong**[1]* **Chuan Fang**[2] **Jia Zheng**[1]
Rui Tang[1] **Hao Zhu**[1] **Ping Tan**[2] **Zihan Zhou**[1]
[1]Manycore Tech Inc. [2]Hong Kong University of Science and Technology
`https://manycore-research.github.io/SpatialLM`

Abstract

SpatialLM is a large language model designed to process 3D point cloud data and generate structured 3D scene understanding outputs. These outputs include architectural elements like walls, doors, windows, and oriented object boxes with their semantic categories. Unlike previous methods which exploit task-specific network designs, our model adheres to the standard multimodal LLM architecture and is fine-tuned directly from open-source LLMs.

To train SpatialLM, we collect a large-scale, high-quality synthetic dataset consisting of the point clouds of 12,328 indoor scenes (54,778 rooms) with ground-truth 3D annotations, and conduct a careful study on various modeling and training decisions. On public benchmarks, our model gives state-of-the-art performance in layout estimation and competitive results in 3D object detection. With that, we show a feasible path for enhancing the spatial understanding capabilities of modern LLMs for applications in augmented reality, embodied robotics, and more.

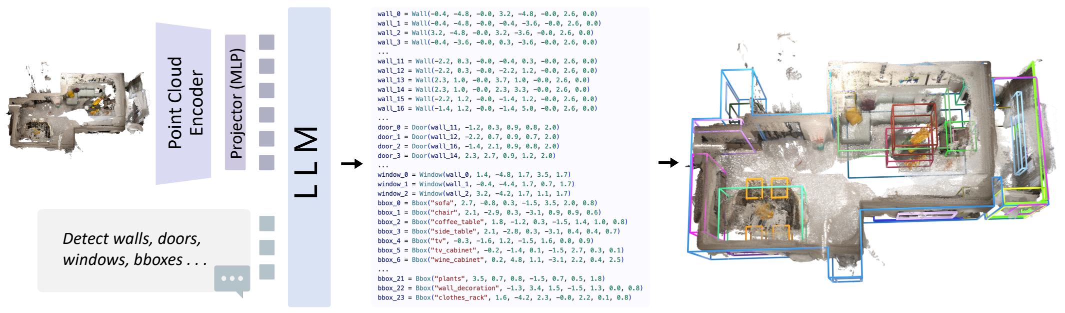
1 Introduction

3D indoor environments are ubiquitous in our daily lives. We spend a lot of time and perform various activities in such environments every day. Therefore, a long-standing goal of artificial intelligence is to teach machines to perceive, reason about, and interact with 3D indoor scenes as humans do. In this work, we focus on the task of *structured indoor modeling*, which aims to extract structured indoor scene descriptions from raw sensorial inputs (*i.e.*, RGBD scans). Specifically, the scene description includes both the architectural layout (*i.e.*, walls, doors, and windows) and 3D object bounding boxes in the indoor environment. Such 3D structure information has been shown to benefit a wide range of real-world applications such as scene editing [40], augmented reality [4, 2], and robot navigation [17].

Compared to 3D formats such as meshes, voxels, or implicit functions, the structured scene description provides a highly compact yet flexible scene representation. In this paper, our *first contribution* is to regard the structured descriptions as scripts of a general-purpose language (*i.e.*, Python) and propose to predict the language in text form, as illustrated in Figure 1. This design choice has several advantages: (i) it is human interpretable and editable; (ii) it can be easily extended to incorporate any new classes without affecting existing content in the scripts; and (iii) it allows us to leverage the strong built-in coding capability of pre-trained large language models.

Therefore, our goal is to implement our method by directly fine-tuning open-source LLMs. While modern (multimodal) LLMs have revolutionized fields like natural language processing, 2D image understanding and generation, few work has attempted to use LLMs for 3D structured scene modeling. This is partly due to the lack of large-scale, high-quality datasets for the task. In this paper, as our *second contribution*, we collect a new synthetic dataset, which consists of the point clouds of 12,328

*Equal contribution.



```
wall_0 = Wall(-0.4, -4.8, -0.0, 3.2, -4.8, -0.0, 2.6, 0.0)
wall_1 = Wall(-0.4, -4.8, -0.0, -0.4, -3.6, -0.0, 2.6, 0.0)
wall_2 = Wall(3.2, -4.8, -0.0, 3.2, -3.6, -0.0, 2.6, 0.0)
wall_3 = Wall(-0.4, -3.6, -0.0, 0.3, -3.6, -0.0, 2.6, 0.0)
...
wall_11 = Wall(-2.2, 0.3, -0.0, -0.4, 0.3, -0.0, 2.6, 0.0)
wall_12 = Wall(-2.2, 0.3, -0.0, -2.2, 1.2, -0.0, 2.6, 0.0)
wall_13 = Wall(2.3, 1.0, -0.0, 3.7, 1.0, -0.0, 2.6, 0.0)
wall_14 = Wall(2.3, 1.0, -0.0, 2.3, 3.3, -0.0, 2.6, 0.0)
wall_15 = Wall(-2.2, 1.2, -0.0, -1.4, 1.2, -0.0, 2.6, 0.0)
wall_16 = Wall(-1.4, 1.2, -0.0, -1.4, 5.0, -0.0, 2.6, 0.0)
...
door_0 = Door(wall_11, -1.2, 0.3, 0.9, 0.8, 2.0)
door_1 = Door(wall_12, -2.2, 0.7, 0.9, 0.7, 2.0)
door_2 = Door(wall_16, -1.4, 2.1, 0.9, 0.8, 2.0)
door_3 = Door(wall_14, 2.3, 2.7, 0.9, 1.2, 2.0)
...
window_0 = Window(wall_0, 1.4, -4.8, 1.7, 3.5, 1.7)
window_1 = Window(wall_1, -0.4, -4.4, 1.7, 0.7, 1.7)
window_2 = Window(wall_2, 3.2, -4.2, 1.7, 1.1, 1.7)
bbox_0 = Bbox("sofa", 2.7, -0.8, 0.3, -1.5, 3.5, 2.0, 0.8)
bbox_1 = Bbox("chair", 2.1, -2.9, 0.3, -3.1, 0.9, 0.9, 0.6)
bbox_2 = Bbox("coffee_table", 1.8, -1.2, 0.3, -1.5, 1.4, 1.0, 0.8)
bbox_3 = Bbox("side_table", 2.1, -2.8, 0.3, -3.1, 0.4, 0.4, 0.7)
bbox_4 = Bbox("tv", -0.3, -1.6, 1.2, -1.5, 1.6, 0.0, 0.9)
bbox_5 = Bbox("tv_cabinet", -0.2, -1.4, 0.1, -1.5, 2.7, 0.3, 0.1)
bbox_6 = Bbox("wine_cabinet", 0.2, 4.8, 1.1, -3.1, 2.2, 0.4, 2.5)
...
bbox_21 = Bbox("plants", 3.5, 0.7, 0.8, -1.5, 0.7, 0.5, 1.8)
bbox_22 = Bbox("wall_decoration", -1.3, 3.4, 1.5, -1.5, 1.3, 0.0, 0.8)
bbox_23 = Bbox("clothes_rack", 1.6, -4.2, 2.3, -0.0, 2.2, 0.1, 0.8)
```

Figure 1: The overall pipeline of SPATIALLM. Given the point cloud input, it employs a standard "Encoder-MLP-LLM" architecture for multimodal feature alignment **(left)**, and generates structured scene descriptions in pure text form as output **(middle)**. The reconstructed 3D structure is further overlaid on the point cloud for visualization **(right)**.

distinct scenes (54,778 rooms) paired with 3D structure information, and conduct the first empirical study on the best strategy of aligning the point cloud input with LLMs for structured scene modeling.

Building upon our findings, we introduce SPATIALLM, a large language model which is capable of processing point cloud input and generating structured scene descriptions. As our *third contribution*, we show that, by first training on our large-scale dataset and then on smaller downstream task data, our model gives competitive performance on public benchmarks for layout estimation and 3D object detection, respectively. Additionally, we provide proof-of-concept zero-shot experiments which show that our model can handle point clouds from diverse sources such as monocular video sequences. These results suggest that SPATIALLM may serve as a base for developing future solutions for scenarios where enhanced spatial understanding and reasoning is required (*e.g.*, embodied AI).

2 Related Work

2.1 Structured Indoor Modeling

Structured indoor modeling aims to recover structural elements (*i.e.*, walls, doors, and windows) as well as 3D objects from unstructured point clouds. In the literature, some studies focus on room layout estimation from point clouds. Early methods [37, 38] rely on traditional geometric analysis algorithms (*i.e.*, RANSAC) to detect wall planes. SceneCAD [1] jointly optimizes the layout estimation and CAD model alignment of the scanned scene. RoomFormer [60] proposes to train a Transformer-based network to predict corners and rooms. Meanwhile, a number of recent methods [35, 46, 51, 47, 25] have advanced the state-of-the-art on 3D object detection from point clouds.

The closest work to ours is SceneScript [2], which also casts structured indoor reconstruction as a sequence modeling problem. It demonstrates great flexibility as the language can be tailored to different scenarios without changing the method. But unlike our method, SceneScript uses domain-specific tokens for language commands, thus requiring a specialized decoder. Instead, SPATIALLM strictly adheres to the "Encoder-MLP-LLM" architecture and is directly fine-tuned based on modern foundation models.

2.2 Large Language Models for 3D Scene Understanding

Recently, there has been a growing interest in empowering LLMs to understand and reason about 3D spaces, as defined in public benchmarks such as ScanRefer [9] and ScanQA [3]. One line of work [53, 20, 19, 29, 22] first uses 3D detectors to locate objects in the point cloud, then extracts features of individual objects using a pre-trained object-level encoder. In this paper, we study how such 3D detectors can be learned with modern LLMs.

Other methods directly extract scene-level features from the point cloud and align the features with LLMs [18, 66, 10, 16]. 3D-LLM [18] uses pre-trained image encoder to obtain 2D features from multi-view images and back-projects onto the point cloud [21]. It then leverages a Q-former [28] to map them to a fixed number of tokens as input to the LLM. LL3DA [10] adopts a 3D visual encoder

```
@dataclass
class Wall:
    a_x: int
    a_y: int
    a_z: int
    b_x: int
    b_y: int
    b_z: int
    height: int

@dataclass
class Door:
    wall_id: str
    position_x: int
    position_y: int
    position_z: int
    width: int
    height: int

@dataclass
class Window:
    wall_id: str
    position_x: int
    position_y: int
    position_z: int
    width: int
    height: int

@dataclass
class Bbox:
    class: str
    position_x: int
    position_y: int
    position_z: int
    angle_z: int
    scale_x: int
    scale_y: int
    scale_z: int
```

Figure 2: Definition of our structured representation for layouts and objects.

pre-trained on ScanNet detection as the scene encoder. ScanReason [66] further employs a 3D point cloud encoder to obtain fine-grained geometric information for accurate object grounding. Scene-LLM [16] tokenizes the 3D point cloud features by simply dividing the space into a fixed-resolution voxel grid and aggregating features in the same voxel.

Besides point cloud-based representations, recent studies have explored other ways to integrate 3D information in LLMs. SpatialVLM [8] primarily focuses on synthesizing large-scale 3D VQA data to enhance the spatial reasoning capabilities of the VLM. LLaVA-3D [67] injects 3D position embeddings into 2D CLIP features to establish a unified architecture for 2D image understanding and 3D scene understanding. SpatialRGPT [11] extracts a 3D scene graph from a single image to enhance grounded spatial reasoning in LLMs. Video-3D LLM [63] learns representations from video frames and 3D coordinates to leverage the strong priors of pre-trained 2D Video LLMs. LSceneLLM [65] uses an adaptive self-attention module to capture fine-grained details in local areas. Our work differs from these studies in that we focus on analyzing the structural information of the scene.

3 SpatialLM

Given the point cloud **P** (typically derived from RGBD scans), our goal is to recover the 3D layout **L**, which consists of all walls, doors and windows, and a set of objects **O** in the scene. As discussed before, we represent **L** and **O** as scripts of a general-purpose language, and fine-tune LLMs to predict the language in text form. Figure 2 shows all the parameters we use for the layout and objects.

To the best of our knowledge, we are the first to employ LLMs for structured scene reconstruction from point clouds. Thus, we first propose a new dataset suitable for training the LLMs on (Section 3.1). Then, with the new dataset, we set out to investigate the following two key questions: (i) What point cloud features to use for the structured indoor modeling task (Section 3.2)? (ii) How to integrate the features into LLMs (Section 3.3)?

3.1 Dataset and Metrics

Table 1 provides a list of popular indoor RGBD scan datasets. We can make a few observations:

First, the number of scenes in real datasets is relatively small, ranging from about one hundred to a few thousands. To collect such a dataset, one needs to take RGBD scans in real scenes and manually annotate 3D information, which is a highly laborious process. Further, most real datasets provide annotations on the objects (*i.e.*, semantic labels and 3D boxes) only. One exception is SceneCAD [1], which manually adds wall annotations to ScanNet dataset. But it only contains 1,151 rooms.

Table 1: Quantitative statistics of indoor RGBD scan datasets.

Dataset (year)	source	#scenes	annotations layouts	annotations objects
ScanNet (2017) [13]	real	1,513		●
Matterport3D (2017) [7]	real	90		●
3RScan (2019) [50]	real	1,482		●
SceneCAD (2020) [1]	real	1,151	●	●
ARKitScenes (2021) [4]	real	5,048		●
MultiScan (2022) [34]	real	273		●
ScanNet++ (2023) [59]	real	1,006		●
HM3DSem (2023) [58]	real	181		●
Structured3D (2020) [64]	syn.	3,500	●	
Hypersim (2021) [45]	syn.	461		●
ProcTHOR (2022) [14]	syn.	12,000	●	●
HSSD (2024) [14]	syn.	211	●	●
ASE (2024) [2]	syn.	100,000	●	
SpatialLM dataset (ours)	syn.	12,328	●	●

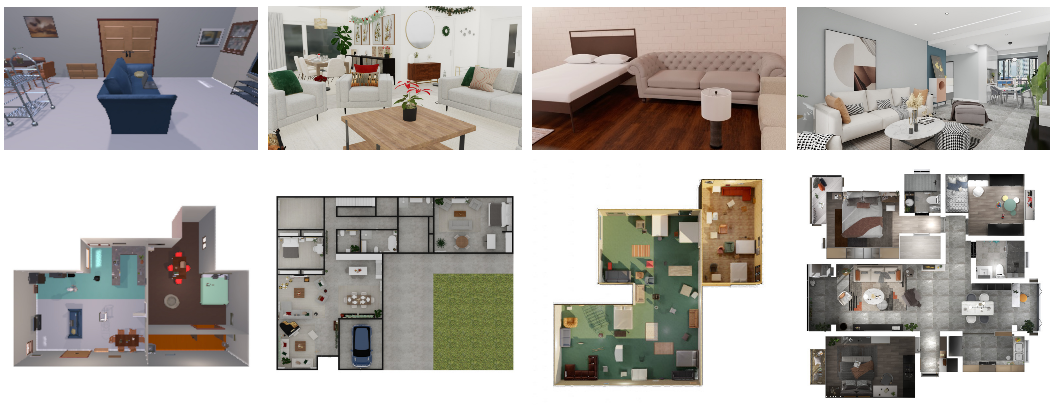


Figure 3: Dataset visual quality comparison. The layout and object placements in ProcTHOR [14] and ASE [2] are program-generated, which exhibit noticeable differences from real-world statistics. The scenes in HSSD [24] and our dataset are fully human-authored. But HSSD only has 211 scenes.

Second, for synthetic datasets, it is much cheaper to obtain ground-truth 3D annotations. But to ensure the realism of the RGBD scans, the 3D scenes must be professionally constructed. As a result, the diversity of 3D scenes in some datasets is still limited. For example, Hypersim [45] and HSSD [24] only have 461 and 211 scenes, respectively. On the other hand, one may create a large number of scenes by employing a rule-based procedural modeling pipeline, as did in ProcTHOR [14] and ASE [2]. But the quality of the scenes is not as high.

SpatialLM dataset. In this paper, we make an effort to build a dataset suitable for training LLMs for structured indoor scene reconstruction. To this end, we take advantage of access to a large repository of interior designs from an online platform[2] in the interior design industry. Most designs are created by professional designers and used for real-world production. We parse each 3D house into individual rooms and use a few rules to filter the rooms. This results in 12,328 distinct scenes with 54,778 rooms.

For the objects, we keep annotations on a selected set of 59 common categories (excluding wall, door, and window), and filter out small objects with side lengths all $< 15\text{cm}$. In the end, our dataset contains 412,932 annotated object instances of 35,426 unique CAD models.

For each scene, we generate photo-realistic RGBD images utilizing an industry-leading rendering engine. At the time of rendering, a camera trajectory traversing each room is simulated, based on which the images are taken at an interval of $0.5\,\text{m}$. Figure 3 compares the visual quality of our dataset to several recent synthetic datasets. Finally, the dataset is divided into 11,328/500/500 scenes for training/validation/testing.

Evaluation metrics. We evaluate our method on both layout estimation and 3D object detection. For *layout estimation*, let $\hat{\mathbf{L}} = \{\hat{l}_i\}_{i=1}^{\hat{m}}$ and $\mathbf{L} = \{l_i\}_{i=1}^{m}$ denote the set of predicted structural elements and ground truth, respectively. Since each element is a plane segment in 3D space, we compute the 2D intersection-over-union (IoU_{2D}) by projecting each predicted element to the plane of the ground-truth element to find the optimal assignment with the Hungarian algorithm, and report the F1-scores at 0.25 and 0.5 thresholds. For *3D object detection*, we match the predictions $\hat{\mathbf{O}} = \{\hat{o}_i\}_{i=1}^{\hat{n}}$ with the ground truth $\mathbf{O} = \{o_i\}_{i=1}^{n}$ via Hungarian matching. Then, we compute the 3D intersection-over-union (IoU_{3D}) between each matched pair and report the F1-scores at 0.25 and 0.5 thresholds.

3.2 Point Cloud Encoders

Extracting meaningful representations for a point cloud is challenging because of its irregular format. Learning-based approaches to process 3D point clouds can be roughly classified into three groups: *Mapping-based methods* are inspired by the success of 2D image backbones such as CLIP [44] and the DINO series [6, 39], and map pixel-aligned image features to 3D. For example, Lexicon3D [33]

[2]`https://www.kujiale.com/`

Table 2: Experiment on the encoder design.

Encoder	Params (M)	F1 - layout IoU$_{2D}$@0.25	F1 - layout IoU$_{2D}$@0.5	F1 - object IoU$_{3D}$@0.25	F1 - object IoU$_{3D}$@0.5
Voxelize ($\mathbf{P}$+DINOv2)	0.0	10.8	5.5	0.1	0.0
Rand. sampling ($\mathbf{P}$+DINOv2)	0.0	10.6	4.9	0.2	0.0
3DCNN enc. ($\mathbf{P}$)	10.5	79.4	76.3	59.8	46.1
3DCNN enc. ($\mathbf{P}$+DINOv2)	11.6	81.5	79.4	57.1	45.0
3DCNN enc. ($\mathbf{P}$) + Voxelize (DINOv2)	11.9	83.8	81.6	62.9	46.7
Sonata/PTv3 enc. ($\mathbf{P}$)	108.8	**84.6**	**82.6**	**65.1**	**49.4**

projects 3D points onto the image plane to obtain their corresponding pixel features; ConceptFusion [21] builds multimodal 3D maps via traditional SLAM and multi-view fusion approaches. *Voxel-based methods* first transform irregular point clouds to regular representations via 3D voxelization. Then, sparse convolution algorithms [12] can be readily used for efficient computation. *Point-based methods*, by contrast, process point clouds directly as sets embedded in continuous space. Recent approaches in this category [62, 54–56] have successfully explored the self-attention mechanism in Transformers and demonstrated superior performance on downstream applications.

Note that all methods discussed above produce one feature per point by default. Given that a scene-level point cloud consists of hundreds of thousands of points or more, these features are too much in terms of quantity for LLMs to consume directly. Since most modern open-source LLMs take an "Encoder-MLP-LLM" architecture for multimodal feature alignment (*e.g.*, the Llama series [32] and Qwen series [42, 43]), we can define a point cloud encoder for LLMs as a mapping which takes a set of N points as input and generates K feature embeddings, where $K \ll N$:

$$\mathbf{F} = \mathcal{E}(\mathbf{P}), \quad \mathbf{P} \in \mathbb{R}^{N\times 6}, \mathbf{F} \in \mathbb{R}^{K\times D}. \tag{1}$$

Here, each input point is a 6-dimensional vector (XYZ and RGB), and D is the feature dimension.

Recently, a few attempts have been made to convert point representations into feature embeddings [2, 16]. However, this step has not been carefully investigated. In this section, we conduct experiments to study (i) the effectiveness of different encoders, and (ii) the impact of K, *i.e.*, the number of output features. Unless otherwise stated, Qwen2.5-0.5B [42] is employed as our base model. We use Sonata [56] as the point cloud encoder, and a two-layer MLP as the projector.

Experiment on the encoder design. We consider six possible point cloud encoders. The first group of encoders adopts a mapping-based method to fuse 2D image features to 3D points. Following Lexicon3D [33], we pool pixel-level DINOv2 features to form 3D point representations. Then, one of the two routes may be taken to reduce the points into a manageable length of tokens:

- **Voxelize (P+DINOv2)**: divides the space into a fixed-resolution voxel grid and aggregates all the points in the same voxel – similar to the approach used in Scene-LLM [16];
- **Rand. sampling (P+DINOv2)**: uses Farthest Point Sampling [41] to randomly sample a fixed number of points with their features.

From Table 2 we can see, these two options result in extremely low F1-scores. We hypothesize that this is because too much spatial information is lost during the naive down-sampling step. While this may be acceptable for certain applications, such as visual Q&A, it is problematic for our task, which relies heavily on the spatial information to reconstruct the geometric coordinates of objects.

The second group of encoders adopts a standard 3DCNN encoder to generate pooled features. We follow SceneScript [2] to build our encoder with a sparse 3D convolution library [48]. Here we consider three variants:

- **3DCNN enc. (P)**: takes raw points (6-dim XYZRGB values) as input;
- **3DCNN enc. (P+DINOv2)**: takes the points with the associated DINOv2 features as input;
- **3DCNN enc. (P) + Voxelize (DINOv2)**: concatenates the output of 3DCNN with aggregated DINOv2 features.

As shown in Table 2, a learned encoder is effective in keeping the necessary semantic and geometric information for reconstruction. We also find that incorporating DINOv2 features yields slightly better results, possibly due to the enhanced contextual information.

The last encoder we consider is **Sonata** [56], a variant of Point Transformer V3 (PTv3) [55] which removes the decoder and focuses on self-supervised learning on the encoder. Compared to other backbones, such an "encoder-only" design makes it quite convenient for adapting to LLMs. Indeed, as shown in Table 2, with raw points as input, it already achieves the best results among all encoders.

Experiment on the spatial resolution. Recall in Eq. (1) that the parameter K denotes the number of feature embeddings or visual tokens, which is critical to the performance of LLMs. In this experiment, we study the effect of K.

However, fixing K is difficult because different point clouds generally produce varying K values. Instead, since the encoders employ a coarse-to-fine structure for point cloud processing, we set the resolution at the finest level at 5cm and use a 5-level hierarchical structure by default. A detailed illustration of the point cloud encoder hierarchy is presented in the appendix (see Figure 11). Then, we gradually increase the resolution up to $4\times$, and report the results in Table 3. As can be seen, model performance steadily improves from $1\times$ to $2\times$ resolutions, but further increasing the resolution hurts the performance. This is likely due to the excessively long token sequences.

Table 3: Experiment on the spatial resolution. Avg. (K) indicates the average value of K under the corresponding resolution.

Res.	Avg. (K)	F1 - layout		F1 - object	
		IoU_{2D}@0.25	IoU_{2D}@0.5	IoU_{3D}@0.25	IoU_{3D}@0.5
$1\times$	123	84.6	82.6	65.1	49.4
$1.25\times$	197	84.8	82.6	65.3	52.2
$1.5\times$	286	86.9	85.1	68.7	58.6
$2\times$	510	**87.4**	**85.3**	71.6	61.1
$2.5\times$	788	86.1	83.9	**72.2**	**61.7**
$4\times$	2,000	85.8	83.7	69.9	58.9

3.3 Training Schedules

As most recent MLLMs start from pre-trained unimodal backbones, how to align the multimodal content with the language model plays a critical role in the performance of the resulting MLLMs. For example, a common practice adopted by many MLLMs [30, 49] is the inclusion of a two-stage training schedule: (i) an alignment stage that trains the randomly initialized MLP projector only, followed by (ii) a fine-tuning stage that trains both the projector and language model while keeping the vision encoder frozen. Other studies (*e.g.*, [23]) advocate a single-stage training protocol, suggesting that including the projector training stage is unnecessary. Furthermore, when adapting the LLMs to rich visual content such as long videos, some research [68] has shown that a three-stage pipeline, which progressively unfreezes different components in different stages, improves the performance.

Table 4: Experiment on training schedule. For each training stage, we use three circles to represent the three components in the "Encoder-MLP-LLM" architecture. "○" and "●" indicate whether a component is frozen or trainable, respectively.

Training stages			F1 - layout		F1 - object	
1st	2nd	3rd	IoU_{2D}@0.25	IoU_{2D}@0.5	IoU_{3D}@0.25	IoU_{3D}@0.5
○●●	-	-	73.8	68.4	40.2	22.7
●●●	-	-	84.6	82.6	**65.1**	**49.4**
●●○	○●●	-	70.9	62.7	27.4	13.7
●●○	●●●	-	**85.3**	**83.4**	60.7	45.6
○●○	●●○	○●●	76.9	70.1	31.1	16.0
○●○	●●○	●●●	83.2	80.3	57.1	41.2

In this experiment, we test six possible training configurations, ranging from one-stage to three-stage, to evaluate the influence of different training strategies. As shown in Table 4, we found that a single-stage fine-tuning yields the best results. Adding more stages generally degrades the model performance. We also found that making all parameters trainable in a single stage is important, especially for 3D object detection. This may indicate that, compared to modern image encoders such as SigLIP [61] and DINOv2 [39], current pre-trained point cloud encoders are not as versatile in supporting downstream tasks yet.

4 Experiments

In this section, we compare SPATIALLM to the current state-of-the-art on scene layout estimation (Section 4.1) and 3D object detection (Section 4.2), respectively. Following our empirical study, we use Qwen2.5-0.5B [42] as the base model and Sonata [56] as the point cloud encoder. We set of resolution at the finest level at $2.5\mathrm{cm}$ and employ a single-stage training on our dataset.

Besides testing on established RGBD benchmarks, we also investigate whether our model can handle inputs from other sources, such as point clouds reconstructed from RGB videos (Section 4.3). Other extensions and more technical details can be found in the appendix.

4.1 Layout Estimation

Layout estimation focuses on predicting architectural elements, *i.e.*, walls, doors, and windows, within an indoor scene. In this experiment, we include results from two representative baselines, namely RoomFormer [60] and SceneScript [2]. As the state-of-the-art specialist model for layout estimation, RoomFormer employs a highly specialized network architecture, using two-level queries to predict room polygons and their corners, respectively. SceneScript, on the other hand, casts 3D structure estimation as an auto-regressive “next-token-prediction” problem. However, unlike our method, it uses specialized structured language commands to describe the scene and trains a Transformer decoder from scratch.

Following the setting of RoomFormer, we perform evaluation on the Structured3D benchmark [64], which contains 3,500 residential houses with diverse floorplans. We use the original data split of 3000/250/250 for training/validation/testing, respectively. For RoomFormer, we directly download the model checkpoint from its website[3] for inference. Note that RoomFormer takes in 2D density maps projected from point clouds and outputs 2D layouts, which are converted to 3D by extruding based on the lowest/highest points in the point cloud. For SceneScript, we download the pre-trained model and code[4], and fine-tune it on Structured3D with the default hyperparameter settings.

For our model, we include three variants: (i) trained on Structured3D only, (ii) trained on our dataset only, and (iii) first trained on our dataset, then fine-tuned on Structured3D. As can be seen in Table 5, directly training LLMs on the smaller Structured3D dataset leads to poor performance. In contrast, by first training on our new dataset and then on Structured3D, our model outperforms both baselines for this task. This demonstrates the benefits of our new dataset.

Table 5: Experiment on layout estimation.

Method	Params (M)	F1 - Structured3D IoU_{2D}@0.25	F1 - Structured3D IoU_{2D}@0.5
RoomFormer [60]	42.2	83.4	81.4
SceneScript [2]	29.9	90.4	89.2
SPATIALLM (ft. Structured3D)	603.5	32.6	18.1
SPATIALLM (ft. Ours)	603.5	59.9	44.7
SPATIALLM (ft. Ours $\rightarrow$ Structured3D)	603.5	**94.3**	**93.5**

Figure 4 provides some qualitative comparisons. Since RoomFormer detects room elements independently, we can see that some doors and windows are not attached to any wall. Auto-regressive methods like SceneScript and our method can preserve the relationships between structures by design. But we observe that SceneScript tends to miss some elements in its output.

4.2 3D Object Detection

Next, we compare SPATIALLM to existing methods on 3D object detection. We include results from two representative baselines, namely V-DETR [47] and SceneScript. As the state-of-the-art

[3] https://github.com/ywyue/RoomFormer

[4] https://github.com/facebookresearch/scenescript

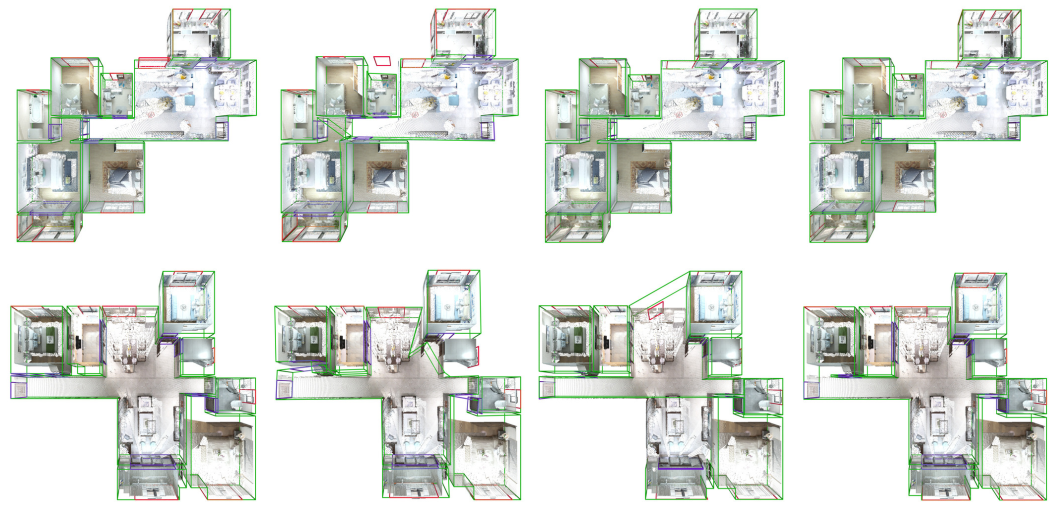


Figure 4: Qualitative results on Structured3D dataset.

specialist model for the task, V-DETR builds upon the DETR framework [5] and makes several task-specific improvements, including a 3D vertex relative position encoding (3DV-RPE), object-based normalization, among others. Following prior practice, we perform evaluation on ScanNet [13], which contains 1,513 3D indoor scans with annotations of 18 object categories. The training set is composed of 1,201 scenes, while 312 scenes are used for validation/testing. Since auto-regressive models (*i.e.*, SPATIALLM and SceneScript) do not produce confidence scores, we report F1 scores instead of mean Average Precision (mAP), as suggested in [2].

For V-DETR, we directly use the model checkpoint downloaded from its website[5]. A confidence threshold of 0.5 is applied for both objectness and semantic predictions, followed by non-maximum suppression (NMS). For SceneScript, we fine-tune the model on ScanNet with the default hyperparameter settings. During fine-tuning, we apply the same data augmentations as used in PTv3 [55] and V-DETR [47].

Table 6: Experiment on 3D object detection. "*": numbers reported on the overlapping subset of object classes between our dataset and ScanNet.

Method	Params (M)	F1 - ScanNet IoU$_{3D}$@0.25	IoU$_{3D}$@0.5
V-DETR [47]	79.4	65.1	**56.8**
SceneScript [2]	29.9	49.1	36.8
SPATIALLM (ft. ScanNet)	603.5	2.9	0.7
SPATIALLM (ft. Ours)	603.5	33.8*	22.6*
SPATIALLM (ft. Ours → ScanNet)	603.5	**65.6**	52.6

For our model, we again include three variants: (i) trained on ScanNet only, (ii) trained on our dataset only, and (iii) first trained on our dataset, then fine-tuned on ScanNet. As can be seen in Table 6, the first variant has very low F1 scores, suggesting that ScanNet itself is too small to train LLMs on. By first training on our new dataset and then on ScanNet, our model outperforms SceneScript and achieves competitive results against the state-of-the-art specialist V-DETR.

Figure 5 shows some qualitative results. Compared to SceneScript, SPATIALLM demonstrates improved detection accuracies across categories. In comparison to V-DETR, we observe that our method has some difficulty in locating objects in "picture", "sink" and "showercurtain" classes. Notably, "picture" and "sink" are the smallest objects in ScanNet by volume ($< 0.1\mathrm{m}^3$), whereas "showercurtain" category is not included in our dataset and has the fewest occurrences in the ScanNet.

[5]https://github.com/V-DETR/V-DETR

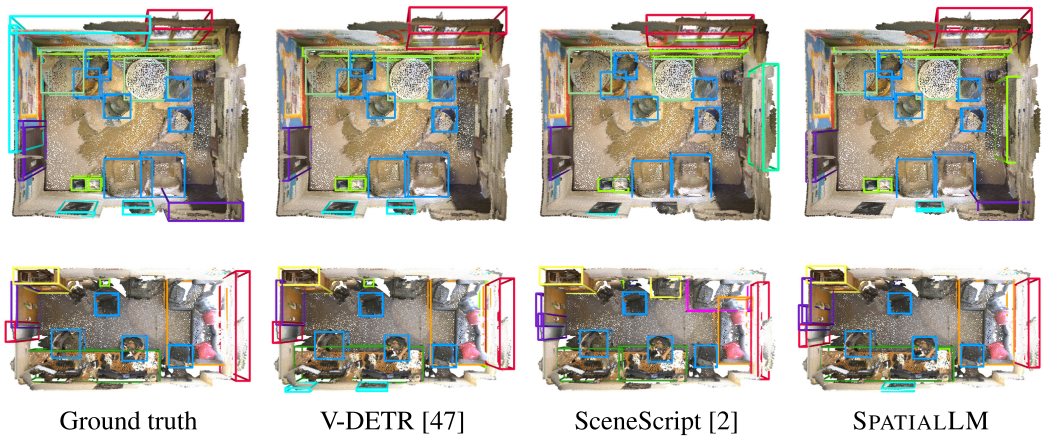


Figure 5: Qualitative results on ScanNet dataset.

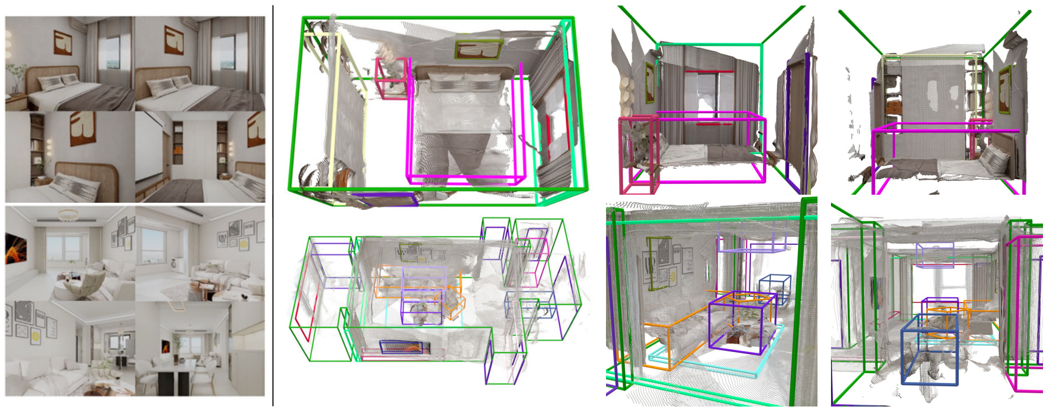


Figure 6: Qualitative results of zero-shot detection on videos.

4.3 Zero-shot Detection on Videos

Although significant progress has been made in SFM and SLAM techniques over the past decades, reliably inferring 3D structure from videos remains a highly challenging problem. Recently, DUSt3R [52] and its extensions [27, 15, 36, 31] provide a novel solution that employs deep networks for simultaneous point matching and camera motion estimation. These techniques are shown to be remarkably robust in dealing with real-world videos or unconstrained image collections.

In this experiment, we collect a set of 107 videos of virtual indoor scene tours and use MASt3R-SLAM [36] to reconstruct a point cloud from each video. Then, we test SpatialLM in a zero-shot setting on the reconstructed point clouds. Unlike RGBD scans, these reconstructions have significant noises, occlusions, and geometric artifacts, posing a greater challenge for structured 3D understanding.

Figure 6 shows some qualitative results. Without fine-tuning, SpatialLM demonstrates strong resilience to imperfect data and preserves consistent layout and object-level predictions. Trained on synthetic datasets with accurate bounding boxes, the model infers full object sizes and orientations from sparse or occluded inputs. In the first example, it predicts beds and nightstands extending to the floor, even when those regions are not visible. Leveraging the auto-regressive nature of LLMs, SpatialLM is also capable of generating plausible room layouts from partial observations. In the second example, the model reasonably reconstructs areas such as the balcony and dining space, filling in missing elements based on contextual understanding.

5 Conclusion

In this study, we demonstrate the feasibility of training LLMs for structured indoor modeling tasks. We regard it as a meaningful step towards building future foundation models that can not only understand, but also reason about, interact with, and even create structured 3D scenes.

Limitations. SPATIALLM has several limitations. *First*, it falls short of delivering a universal, state-of-the-art model for extracting 3D structure from *arbitrary* point clouds. There are significant differences between point clouds from different sources, such as monocular videos, RGBD scans, and LiDAR sensors. As shown in Section 4, although our model exhibits reasonable generalizability across datasets, fine-tuning on a specific dataset is still needed to achieve the best performance. Further scaling up the data and model size could be a promising future direction. *Second*, in this paper, we focus on training LLM for structured indoor modeling only, without considering how it affects the model's natural skills in natural language processing and reasoning. To address this, a more thorough benchmarking may be performed. *Third*, our current approach models indoor layouts using a set of predefined object categories, limiting the LLMs in leveraging their open-ended language capabilities. Future directions include extending our work to support open-vocabulary object detection and 3D visual question answering (VQA), enabling more flexible and generalizable scene understanding.

Impacts. We envision that SPATIALLM will be used in real-world applications such as layout estimation and 3D object detection. For example, people may take our code and fine-tune the model on their own datasets. On the methodology side, we hope that our work will inspire future studies on MLLM models for 3D scene understanding, reasoning, and creation.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Key R&D Program of Zhejiang Province (2025C01001) and the HKUST project 24251090T019.

References

[1] Armen Avetisyan, Tatiana Khanova, Christopher B. Choy, Denver Dash, Angela Dai, and Matthias Nießner. SceneCAD: Predicting object alignments and layouts in RGB-D scans. In *ECCV*, pages 596–612, 2020.

[2] Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan P. Frost, Luke Holland, Campbell Orme, Jakob Engel, Edward Miller, Richard A. Newcombe, and Vasileios Balntas. SceneScript: Reconstructing scenes with an autoregressive structured language model. In *ECCV*, pages 247–263, 2024.

[3] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. ScanQA: 3D question answering for spatial scene understanding. In *CVPR*, pages 19107–19117, 2022.

[4] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitScenes: A diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS Datasets and Benchmarks*, 2021.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9630–9640, 2021.

[7] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*, pages 667–676, 2017.

[8] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas J. Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, pages 14455–14465, 2024.

[9] Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner. ScanRefer: 3D object localization in RGB-D scans using natural language. In *ECCV*, pages 202–221, 2020.

[10] Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. LL3DA: visual interactive instruction tuning for omni-3d understanding, reasoning, and planning. In *CVPR*, pages 26418–26428, 2024.

[11] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded spatial reasoning in vision-language models. In *NeurIPS*, pages 135062–135093, 2024.

[12] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019.

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.

[14] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *NeurIPS*, pages 5982–5994, 2022.

[15] Bardienus Pieter Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jérôme Revaud. MASt3R-SfM: a fully-integrated solution for unconstrained structure-from-motion. In *3DV*, 2025.

[16] Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-LLM: Extending language model for 3d visual reasoning. In *WACV*, pages 2195–2206, 2025.

[17] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *ACL*, pages 7606–7623, 2022.

[18] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3d world into large language models. In *NeurIPS*, pages 20482–20494, 2023.

[19] Haifeng Huang, Yilun Chen, Zehan Wang, Rongjie Huang, Runsen Xu, Tai Wang, Luping Liu, Xize Cheng, Yang Zhao, Jiangmiao Pang, and Zhou Zhao. Chat-Scene: Bridging 3d scene and large language models with object identifiers. In *NeurIPS*, pages 113991–114017, 2024.

[20] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *ICML*, pages 20413–20451, 2024.

[21] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd. Omama, Ganesh Iyer, Soroush Saryazdi, Tao Chen, Alaa Maalouf, Shuang Li, Nikhil Varma Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, K. Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. ConceptFusion: Open-set multimodal 3d mapping. In *RSS*, 2023.

[22] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. SceneVerse: Scaling 3d vision-language learning for grounded scene understanding. In *ECCV*, pages 289–310, 2024.

[23] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the design space of visually-conditioned language models. In *ICML*, 2024.

[24] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation. In *CVPR*, pages 16384–16393, 2024.

[25] Maksim Kolodiazhnyi, Anna Vorontsova, Matvey Skripkin, Danila Rukhovich, and Anton Konushin. UniDet3D: Multi-dataset indoor 3d object detection. In *AAAI*, pages 4365–4373, 2025.

[26] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[27] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with MASt3R. In *ECCV*, pages 71–91, 2024.

[28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742, 2023.

[29] Dingning Liu, Xiaoshui Huang, Yuenan Hou, Zhihui Wang, Zhenfei Yin, Yongshun Gong, Peng Gao, and Wanli Ouyang. Uni3D-LLM: Unifying point cloud perception, generation and editing with large language models. *CoRR*, abs/2402.03327, 2024.

[30] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, pages 34892–34916, 2023.

[31] Yuzheng Liu, Siyan Dong, Shuzhe Wang, Yanchao Yang, Qingnan Fan, and Baoquan Chen. SLAM3R: real-time dense scene reconstruction from monocular RGB videos. In *CVPR*, pages 16651–16662, 2025.

[32] LLama Team. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783, 2024.

[33] Yunze Man, Shuhong Zheng, Zhipeng Bao, Martial Hebert, Liangyan Gui, and Yu-Xiong Wang. Lexicon3D: Probing visual foundation models for complex 3d scene understanding. In *NeurIPS*, pages 76819–76847, 2024.

[34] Yongsen Mao, Yiming Zhang, Hanxiao Jiang, Angel X. Chang, and Manolis Savva. MultiScan: Scalable RGBD scanning for 3d environments with articulated objects. In *NeurIPS*, pages 9058–9071, 2022.

[35] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *ICCV*, pages 2886–2897, 2021.

[36] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *CVPR*, pages 16695–16705, 2025.

[37] Srivathsan Murali, Pablo Speciale, Martin R. Oswald, and Marc Pollefeys. Indoor Scan2BIM: Building information models of house interiors. In *IROS*, pages 6126–6133, 2017.

[38] Sebastian Ochmann, Richard Vock, and Reinhard Klein. Automatic reconstruction of fully volumetric 3d building models from point clouds. *CoRR*, abs/1907.00631, 2019.

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

[40] Akshay Gadi Patil, Supriya Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. Advances in data-driven analysis and synthesis of 3d indoor scenes. *Comput. Graph. Forum*, 43(1), 2024.

[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017.

[42] Qwen Team. Qwen2.5 Technical Report. *CoRR*, abs/2412.15115, 2024.

[43] Qwen Team. Qwen2.5-VL Technical Report. *CoRR*, abs/2502.13923, 2025.

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021.

[45] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, pages 10912–10922, 2021.

[46] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. FCAF3D: fully convolutional anchor-free 3d object detection. In *ECCV*, pages 477–493, 2022.

[47] Yichao Shen, Zigang Geng, Yuhui Yuan, Yutong Lin, Ze Liu, Chunyu Wang, Han Hu, Nanning Zheng, and Baining Guo. V-DETR: DETR with vertex relative position encoding for 3d object detection. In *ICLR*, 2024.

[48] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient point cloud inference engine. In *MLSys*, pages 302–315, 2022.

[49] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Iyer, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, Xichen Pan, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. In *NeurIPS*, pages 87310–87356, 2024.

[50] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. RIO: 3d object instance re-localization in changing indoor environments. In *ICCV*, pages 7657–7666, 2019.

[51] Haiyang Wang, Lihe Ding, Shaocong Dong, Shaoshuai Shi, Aoxue Li, Jianan Li, Zhenguo Li, and Liwei Wang. CAGroup3D: Class-aware grouping for 3d object detection on point clouds. In *NeurIPS*, pages 29975–29988, 2022.

[52] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. DUSt3R: Geometric 3d vision made easy. In *CVPR*, pages 20697–20709, 2024.

[53] Zehan Wang, Haifeng Huang, Yang Zhao, Ziang Zhang, Tao Jin, and Zhou Zhao. Data-efficiently learn large language model for universal 3D scene perception. In *NAACL*, pages 313–333, 2025.

[54] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point Transformer V2: Grouped vector attention and partition-based pooling. In *NeurIPS*, pages 33330–33342, 2022.

[55] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler, faster, stronger. In *CVPR*, pages 4840–4851, 2024.

[56] Xiaoyang Wu, Daniel DeTone, Duncan P. Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob J. Engel, Richard A. Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *CVPR*, pages 22193–22204, 2025.

[57] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, pages 21469–21480, 2024.

[58] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Théophile Gervet, John M. Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, Alexander William Clegg, and Devendra Singh Chaplot. Habitat-matterport 3d semantics dataset. In *CVPR*, pages 4927–4936, 2023.

[59] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *ICCV*, pages 12–22, 2023.

[60] Yuanwen Yue, Theodora Kontogianni, Konrad Schindler, and Francis Engelmann. Connecting the dots: Floorplan reconstruction using two-level queries. In *CVPR*, pages 845–854, 2023.

[61] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, pages 11941–11952, 2023.

[62] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, pages 16239–16248, 2021.

[63] Duo Zheng, Shijia Huang, and Liwei Wang. Video-3D LLM: Learning position-aware video representation for 3d scene understanding. In *CVPR*, pages 8995–9006, 2025.

[64] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A large photo-realistic dataset for structured 3d modeling. In *ECCV*, pages 519–535, 2020.

[65] Hongyan Zhi, Peihao Chen, Junyan Li, Shuailei Ma, Xinyu Sun, Tianhang Xiang, Yinjie Lei, Mingkui Tan, and Chuang Gan. LSceneLLM: Enhancing large 3d scene understanding using adaptive visual preferences. In *CVPR*, pages 3761–3771, 2025.

[66] Chenming Zhu, Tai Wang, Wenwei Zhang, Kai Chen, and Xihui Liu. ScanReason: Empowering 3d visual grounding with reasoning capabilities. In *ECCV*, pages 151–168, 2024.

[67] Chenming Zhu, Tai Wang, Wenwei Zhang, Jiangmiao Pang, and Xihui Liu. LLaVA-3D: A simple yet effective pathway to empowering lmms with 3d-awareness. *CoRR*, abs/2409.18125, 2024.

[68] Orr Zohar, Xiaohan Wang, Yann Dubois, Nikhil Mehta, Tong Xiao, Philippe Hansen-Estruch, Licheng Yu, Xiaofang Wang, Felix Juefei-Xu, Ning Zhang, Serena Yeung-Levy, and Xide Xia. Apollo: An exploration of video understanding in large multimodal models. In *CVPR*, pages 18891–18901, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims presented in the abstract and introduction are well-supported by the paper's contributions. We use a figure (Figure 1) to illustrate the technical scope of SPATIALLM. In the subsequent sections, we introduce a new dataset and conduct comprehensive experiments to support these claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all necessary details to enable reproduction of our experimental results, including dataset construction, evaluation metrics, and training settings in Sections 3 and 4. Data augmentation strategies, hyperparameters, and additional training configurations are included in the supplementary material. Our dataset and model will be made publicly available to support reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have made our code, model, and dataset publicly available. All resources are available at `https://manycore-research.github.io/SpatialLM/`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We clearly describe the dataset curation process and data splits in Section 3.1. The splits used for public datasets, along with details about the checkpoints for each method and evaluation settings, are provided in Sections 4.1 and 4.2. Additional information on data augmentation, hyperparameters, and training configurations is included in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We use well-established metrics for all experiments in this paper. Specifically, we report the IoU_{2D} at 0.25 and 0.5 thresholds for layout estimation, and IoU_{3D} at 0.25 and 0.5 thresholds for 3D object detection. These metrics are calculated by averaging over all samples in the test set.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details of the computational resources used are provided in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification: We have reviewed the Code of Ethics and confirm that our research conforms with it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We discuss broader impacts of this work in Section 5. The authors do not perceive any negative societal impacts of the work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide links to the source codes used in the paper, and to the online platform from which our dataset is collected. The license and terms of use are properly followed.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The dataset curation process is clearly described in Section 3.1.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A SpatialLM Dataset

SpatialLM dataset comprises 12,328 distinct scenes with 54,778 rooms, featuring a total of 403,291 walls, 123,301 doors, and 48,887 windows, with an overall floor area of approximately 863,986 m^2.

Object categories. We have curated a set of 59 commonly occurring object categories, organized by functional and semantic similarity, as shown in Table 7. Statistics of these categories can be found in Figure 7.

Table 7: Categories in the SpatialLM dataset.

Super Categories	**Categories**
Seatings	sofa, chair, dining_chair, bar_chair, stool
Beddings	bed, pillow
Cabinetry	wardrobe, nightstand, tv_cabinet, wine_cabinet, bathroom_cabinet, shoe_cabinet, entrance_cabinet, decorative_cabinet, washing_cabinet, wall_cabinet, sideboard, cupboard
Tables	coffee_table, dining_table, side_table, dressing_table, desk
Kitchen appliances	integrated_stove, gas_stove, range_hood, microwave_oven, sink, stove, refrigerator
Bathroom fixtures	hand_sink, shower, shower_room, toilet, tub
Lighting	illumination, chandelier, floor_standing_lamp
Decoration	wall_decoration, painting, curtain, carpet, plants, potted_bonsai
Electronics	tv, computer, air_conditioner, washing_machine
Others	clothes_rack, mirror, bookcase, cushion, bar, screen, combination_sofa, dining_table_combination, leisure_table_and_chair_combination, multifunctional_combination_bed

Figure 8 plots the object-to-room correlation heatmap, which reveals strong correlations between object distribution and specific room types. Additionally, each room type contains a variety of object categories, highlighting the diversity of our dataset.

Dataset visualizations. We provide visualizations of the ground-truth point cloud and layout annotations in Figure 9 to showcase the quality of the photo-realistic scans and the realism of the layout. Additionally, we show rendered images of objects in Figure 10 to illustrate the diversity and quality of the objects in our dataset.

B Implementation Details of SpatialLM

In Figure 12, we show the full prompt and an example response of SpatialLM. Note that we shift the point cloud and the corresponding layout and object box coordinates to ensure that they are all non-negative. Next, we quantize the coordinates into 1,280 bins, each with a resolution of $2.5\mathrm{cm}$. LLM predicts integer values, which are mapped back to continuous values using the inverse transformation and normalization, restoring them to the original coordinate system.

Data augmentations. We apply a number of augmentations to the point cloud, which are summarized in Table 8 and explained below.

We begin with a cuboid crop following V-DETR [47]. This process involves randomly selecting a point as the center of the cuboid and then determining random dimensions for the cuboid relative to the input point cloud size. We maintain a minimum aspect ratio of 0.8 and ensure that at least 50,000 points remain after the cropping process.

Then, we apply four types of jitter noises to the point cloud: (i) jitter that simulates the minor uncertainties inherent in sensor accuracy; (2) jitter that replicates noise in areas with greater uncertainty; (3) jitter that reflects noise along the edges, typically from floating points between foreground objects and the background; and (4) jitter that represents distant floating points outside the room, often attributed to glossy windows, reflective mirrors, and points along the path to the outdoors.

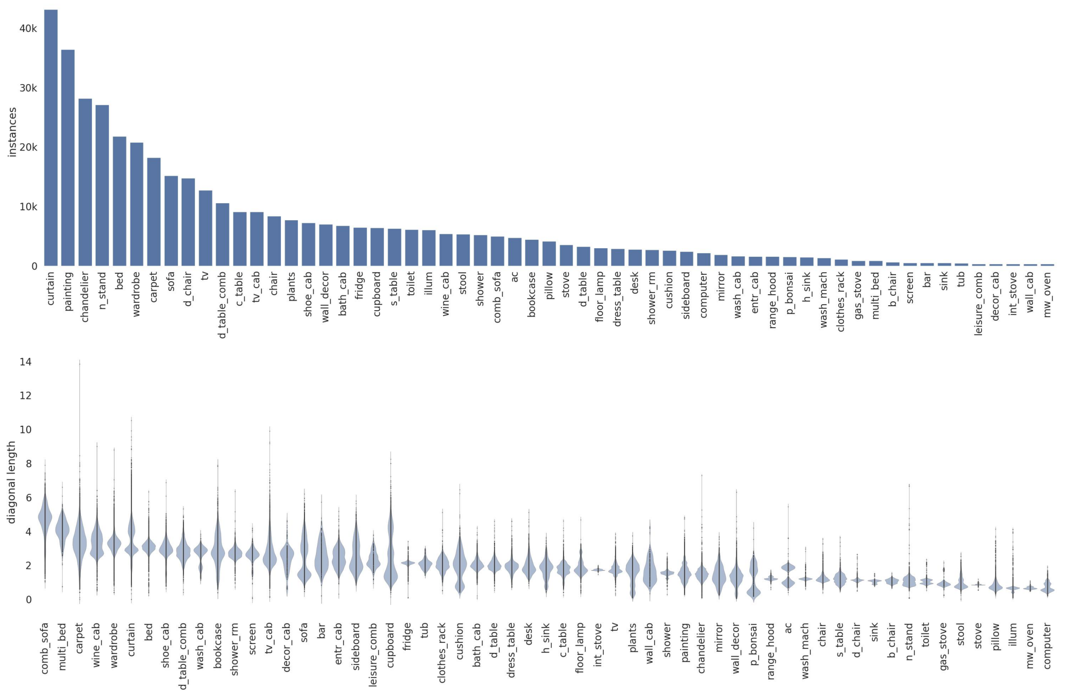


Figure 7: Statistics of the SpatialLM dataset. **Top:** Number of objects in each category. **Bottom:** Distribution of physical sizes in each category. We compute the object size as the diagonal length of the object's bounding box (in meters). All objects in our dataset have realistic physical sizes.

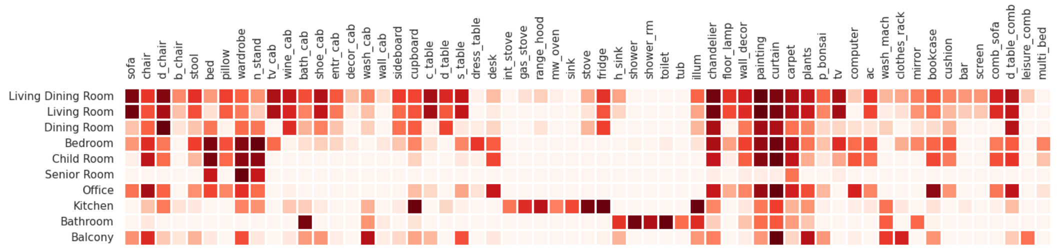


Figure 8: Room object correlation of SpatialLM dataset.

Compute resources. We train SpatialLM for 4 epochs with a total batch size of 64. The learning rate is set at 10^{-4}, using a cosine scheduler with a warm-up ratio of 0.03. The parameters for AdamW optimizer are as follows: adam_beta1 is 0.9, adam_beta2 is 0.99, and adam_epsilon is 1×10^{-8}. We utilized 32 NVIDIA H20 GPUs, and training on SpatialLM dataset takes approximately one day.

For experiments on Structured3D [64] and ScanNet [13], we use a batch size of 8. Fine-tuning on Structured3D involves 50 epochs, while keeping the other settings the same. For fine-tuning on ScanNet, we introduce an additional pre-training stage to mitigate the difference in label spaces across datasets. Specifically, we map the classes in our dataset to those of ScanNet, and further train the model on our dataset for 3 epochs, followed by fine-tuning on ScanNet for 30 epochs.

C Supplementary Results for SpatialLM Experiments

In this section, we present per-category quantitative results for (i) layout estimation, (ii) 3D object detection, and (iii) zero-shot detection on videos.

Table 8: Point cloud data augmentations.

Method	Parameters
cuboid crop	min_points=50,000, aspect=0.8, min_crop=0.75, max_crop=1.0, p=0.1
random jitter	σ=0.025, clip=0.05, ratio=0.8, p=0.9
random jitter	σ=0.2, clip=0.2, ratio=0.05, p=0.8
random jitter	σ=0.4, clip=1.0, ratio=0.001, p=0.75
random jitter	σ=0.5, clip=4.0, ratio=0.0005, p=0.65
elastic distort	params=[[0.2, 0.4], [0.8, 1.6]], p=[0.8, 0.5]
random rotate	axis=z, angle=[0, 90, 180, 270]
random scale	scale=[0.75, 1.25]
auto contrast	p=0.2
chromatic translation	p=0.75, ratio=0.1
chromatic jitter	std=0.05; p=0.8
color drop	p=0.1

C.1 Layout Estimation

Per-category results. In Table 9, we report the F1 scores in percentages at 0.25 and 0.5 thresholds of IoU_{2D} for architectural elements *i.e.*, walls, doors and windows, and the averaged score across the categories on Structured3D test set.

Table 9: Experiment results on layout estimation.

Method	IoU_{2D}@0.25				IoU_{2D}@0.5			
	avg.	wall	door	window	avg.	wall	door	window
RoomFormer [60]	83.4	86.0	85.1	78.9	81.4	84.6	83.1	76.6
SceneScript [2]	90.4	92.9	92.8	85.5	89.2	91.9	92.5	83.3
SPATIALLM (ft. Structured3D)	32.6	54.1	19.4	24.3	18.1	33.5	9.4	11.5
SPATIALLM (ft. Ours)	59.9	71.5	42.7	65.6	44.7	64.8	39.7	29.6
SPATIALLM (ft. Ours $\rightarrow$ Structured3D)	**94.3**	**95.3**	**95.9**	**91.7**	**93.5**	**94.5**	**95.7**	**90.2**

C.2 3D Object Detection

Per-category results. We report the F1 scores in percentages at 0.25 and 0.5 thresholds of IoU_{3D} for 18 ScanNet categories and the averaged score across the categories in Table 10 and Table 11, respectively.

For IoU_{3D}@0.25, we have +16.5% overall gain compared to SceneScript, and +0.5% difference compared to V-DETR. For IoU_{3D}@0.5, we have +15.8% overall gain compared to SceneScript, and -4.2% difference compared to V-DETR. The largest gaps between our model and V-DETR correspond to "picture", "sink", and "shower_curtain". Note that these are either the smallest or the least occurring objects in ScanNet.

Table 10: 3D object detection results with IoU_{3D}@0.25. †: trained on ScanNet only. *: trained on our dataset only. "n/a" indicates the category is not included in our dataset.

Method	average	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	garbbin.	picture	refrige.	s.curtain	sink	sofa	table	toilet	window
V-DETR [47]	65.1	77.4	**80.1**	48.8	**44.4**	82.5	**55.6**	66.1	**68.0**	62.3	54.0	**47.4**	50.8	74.0	**79.5**	72.7	55.3	**98.0**	54.9
SceneScript [2]	49.1	64.5	71.1	39.7	30.3	81.1	43.4	30.9	53.4	41.7	42.6	11.8	28.2	58.6	48.9	68.3	55.8	77.5	36.5
SPATIALLM†	2.9	6.5	7.0	2.5	1.2	4.0	1.4	2.3	4.8	2.5	0.3	0.0	1.4	2.3	0.9	1.7	4.7	5.0	3.7
SPATIALLM*	33.8	57.0	49.2	24.4	12.2	50.9	n/a	15.3	48.1	n/a	n/a	9.7	23.4	n/a	3.6	38.3	31.4	75.6	n/a
SPATIALLM	**65.6**	**80.6**	79.9	**52.0**	40.0	**86.6**	51.0	**66.8**	62.8	**67.1**	**55.6**	29.7	**53.6**	**81.6**	70.7	**78.9**	**63.9**	95.4	**64.3**

Table 11: 3D object detection results with IoU_{3D}@0.5. †: trained on ScanNet only. *: trained on our dataset only. "n/a" indicates the category is not included in our dataset.

Method	average	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	garbbin.	picture	refrige.	s.curtain	sink	sofa	table	toilet	window
V-DETR [47]	**56.8**	68.8	**75.5**	**50.4**	**39.1**	76.2	**43.6**	**54.1**	52.1	**50.9**	**47.8**	**42.7**	**49.2**	**61.5**	**61.4**	65.4	50.6	88.6	43.9
SceneScript [2]	36.8	54.8	65.9	37.3	21.2	73.7	25.3	16.9	44.5	23.5	30.5	3.0	23.3	19.5	24.9	62.0	50.1	64.2	21.7
SpatialLM†	0.7	2.2	2.0	0.0	0.1	0.6	1.4	0.8	1.7	0.4	0.0	0.0	1.4	0.0	0.0	0.0	0.2	2.0	0.0
SpatialLM*	22.6	37.6	41.0	13.5	6.8	30.7	n/a	1.0	34.3	n/a	n/a	2.6	17.7	n/a	1.1	28.4	19.5	59.0	n/a
SpatialLM	52.6	**74.2**	71.7	47.7	29.5	**79.3**	40.1	53.3	**52.6**	46.6	41.7	10.2	42.9	50.6	40.2	**71.1**	**60.3**	**90.5**	**45.0**

C.3 Zero-shot Detection on Videos

Per-category results. We collect 107 virtual room-tour videos and use MASt3R-SLAM [36] to reconstruct a point cloud from each video. Note that these videos are rendered from virtual scenes created by professional designers, thus ground-truth 3D annotations are available. To calibrate the reconstructed point cloud with ground truth 3D annotations, we align the estimated camera trajectory from MASt3R-SLAM to the ground-truth camera trajectory. Then, we perform zero-shot detection with SpatialLM on the calibrated point clouds.

In Table 12, we report the quantitative results on the top-20 occurring categories in the video test set. Note that this is a challenging task, because the reconstructed point clouds are often noisy and highly incomplete. Furthermore, because of the differences in scale and position due to imperfect alignment of camera trajectories, the objects in the point cloud may not match well with the ground truth 3D annotations, leading to lower F1 scores.

Table 12: Zero-shot results on layout estimation and 3D object detection with IoU@0.25.

(a) Layout estimation

avg.	wall	door	window
55.7	68.2	47.4	51.4

(b) 3D object detection

average	curtain	nightstand	chandelier	wardrobe	bed	sofa	chair	cabinet	diningtab.	plants	tvcab.	coffeetab.	sidetab.	aircond.	dresser	stool	refrige.	painting	carpet	tv
36.9	37.0	67.0	36.8	39.6	95.2	69.1	32.3	11.2	24.2	26.3	27.3	64.9	9.7	24.0	46.7	30.8	16.7	38.2	24.1	18.0

D Extensions of SpatialLM

In this section, we present more proof-of-concept experiment results on future extensions of SpatialLM, including (i) handling point clouds from diverse sources, and (ii) adapting to various downstream tasks via language-based prompts.

D.1 Supporting Point Clouds from Diverse Sources

Point clouds offer a lightweight and flexible 3D representation that can be readily generated from a variety of sources. Meanwhile, our dataset is large and diverse, enabling strong zero-shot performance over point cloud inputs in both synthetic and real-world data. In this experiment, we examine the following four types of input sources:

- **Text-to-3D**: We first generate an image via Flux-dev [26] with prompt "a low poly 3D living room in cartoon style", then convert the image into 3D using TRELLIS [57].

- **Hand-held video camera**: We take a real-world video using a hand-held camera and perform 3D reconstruction using MASt3R-SLAM [36].
- **LiDAR-based reconstruction**: We take point clouds reconstructed from captures using iPhone ARKit from MultiScan [34].
- **Synthetic mesh**: We create point clouds by randomly sampling mesh surfaces from a synthetic dataset HSSD [24].

As can be seen in Figure 13, our model is robust to variations in point cloud origin, structure, and appearance.

D.2 Task Adaptation via Language-based Prompts

A key idea of our work is to represent the 3D scene structures in pure text form. This enables SPATIALLM to be conveniently adapted to different downstream tasks via natural language instruction, without changing the underlying model architecture. Below, we demonstrate two such extensions.

Detection with user-specified categories. We enable the model to perform object detection conditioned on user-specified categories by leveraging the flexibility of LLMs. During training, the input prompt is customized to indicate which object categories to detect. The model then learns to selectively predict oriented bounding boxes (OBBs) for instances matching the specified categories, all within a unified training procedure. This experiment highlights the model's adaptability through prompt-based control. Some qualitative results are provided in Figure 14.

Semantic label completion. Besides natural language prompts, SPATIALLM also supports structured language scripts as input, enabling novel tasks such as semantic label completion. In this task, the model receives object bounding boxes and their poses written in a general-purpose language (*i.e.*, Python), while the object categories are left unspecified. The model then predicts the semantic label of each object based on the spatial layout and the corresponding point cloud data. Note that this problem often arises in real-world design workflows that involve 3D assets with known positions but without semantic labels.

Compared to 3D object detection, this task is relatively easy. Nevertheless, it demonstrates the versatility and potential use of large language models for structured 3D understanding. We show some qualitative results in Figure 15. Our model achieves an overall classification accuracy of 96.8% on the test set of our dataset.

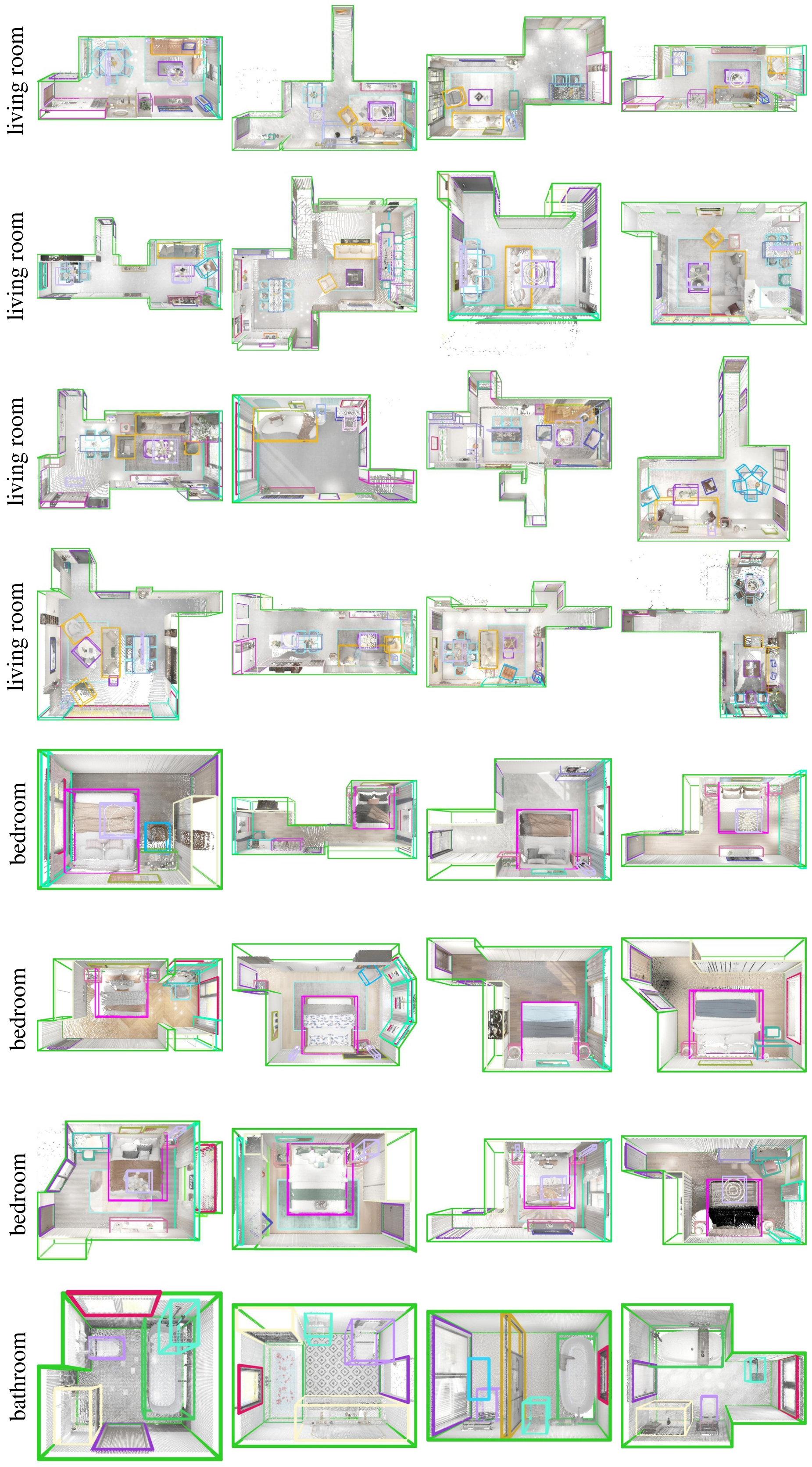


Figure 9: Point clouds with overlaid 3D layout and object annotations in SPATIALLM dataset.

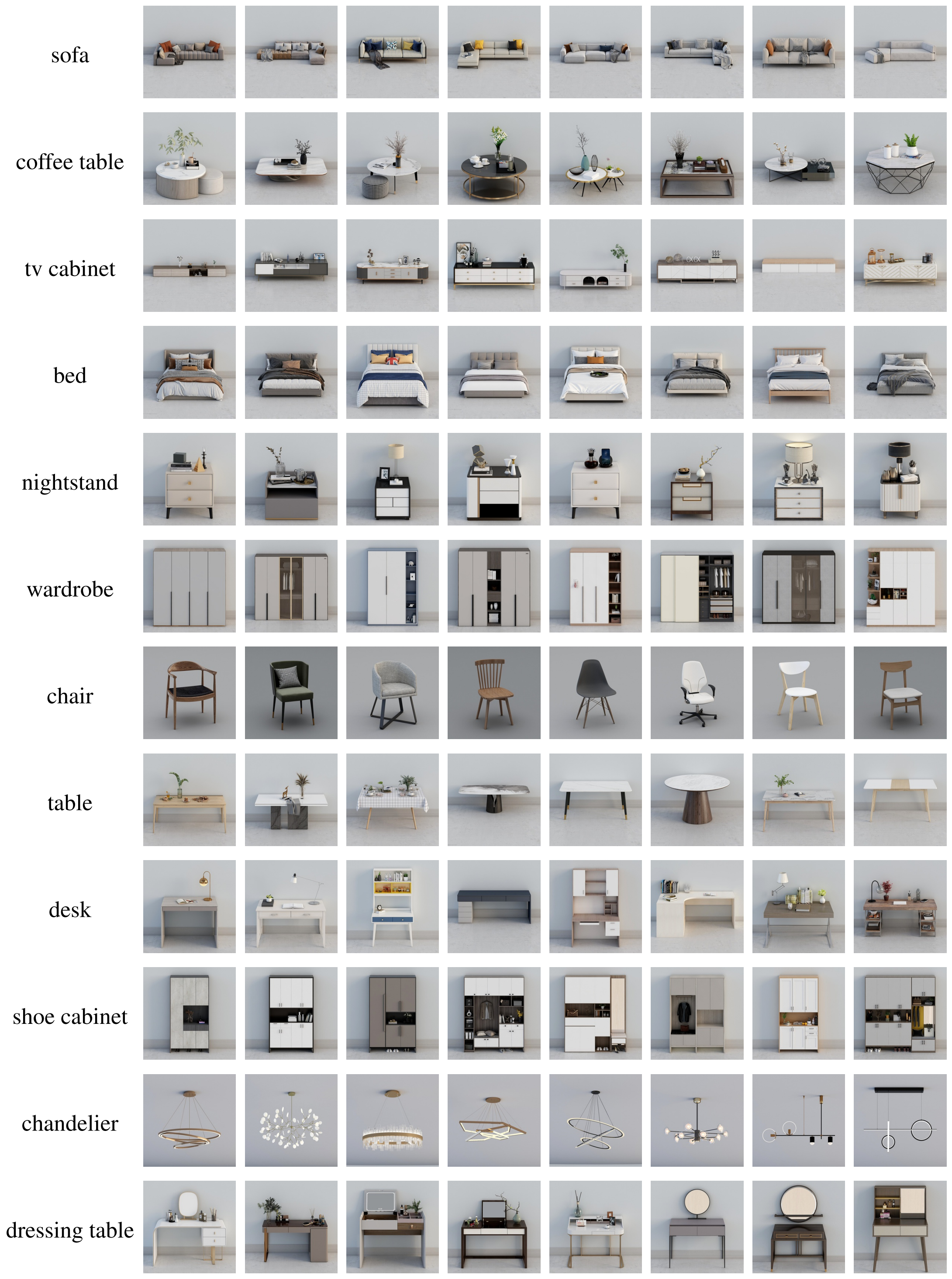


Figure 10: Example objects in SPATIALLM dataset. Note that each category has a diverse set of objects with high quality in both geometry and appearance.

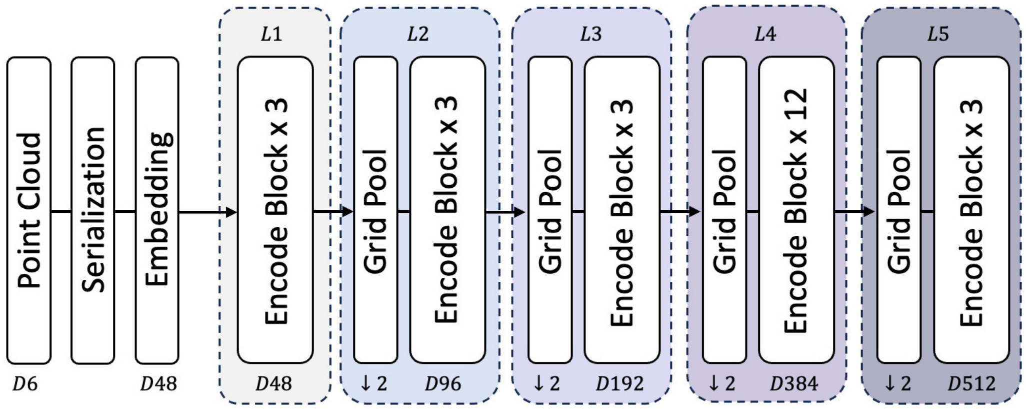
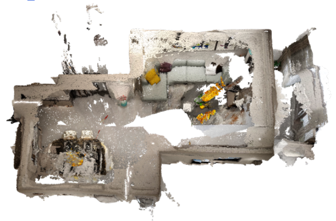


Figure 11: Five-level hierarchical structure of the point cloud encoder [56]. D indicates the feature dimension at each level, and grid pool downscales the point cloud resolution by 2 times along each dimension.

Prompt:



<point_cloud> Detect walls, doors, windows, boxes. The reference code is as follows:

```
@dataclass
class Wall:
    a_x: int
    a_y: int
    a_z: int
    b_x: int
    b_y: int
    b_z: int
    height: int
@dataclass
class Door:
    wall_id: str
    position_x: int
    position_y: int
    position_z: int
    width: int
    height: int
@dataclass
class Window:
    wall_id: str
    position_x: int
    position_y: int
    position_z: int
    width: int
    height: int
@dataclass
class Bbox:
    class: str
    position_x: int
    position_y: int
    position_z: int
    angle_z: int
    scale_x: int
    scale_y: int
    scale_z: int
```

Response:

```
wall_0=Wall(144,15,30,139,15,30,101)
wall_1=Wall(144,15,30,144,26,30,101)
wall_2=Wall(139,15,30,128,16,30,101)
wall_3=Wall(128,16,30,118,18,30,101)
wall_4=Wall(47,18,30,118,18,30,101)
wall_5=Wall(47,18,30,47,42,30,101)
wall_6=Wall(144,26,30,152,26,30,101)
wall_7=Wall(152,26,30,152,42,30,101)
wall_8=Wall(47,42,30,67,42,30,101)
wall_9=Wall(67,42,30,67,53,30,101)
wall_10=Wall(138,42,30,152,42,30,101)
wall_11=Wall(138,42,30,138,53,30,101)
wall_12=Wall(49,53,30,67,53,30,101)
wall_13=Wall(49,53,30,49,151,30,101)
wall_14=Wall(138,53,30,152,53,30,101)
wall_15=Wall(152,53,30,152,170,30,101)
wall_16=Wall(19,151,30,49,151,30,101)
wall_17=Wall(19,151,30,19,269,30,101)
wall_18=Wall(118,170,30,152,170,30,101)
wall_19=Wall(118,170,30,118,269,30,101)
wall_20=Wall(19,269,30,118,269,30,101)
door_0=Door(wall_16,32,151,60,30,75)
door_1=Door(wall_17,19,194,60,27,75)
door_2=Door(wall_19,118,213,60,37,75)
door_3=Door(wall_20,98,269,60,30,75)
window_0=Window(wall_4,94,18,74,73,75)
window_1=Window(wall_17,19,162,60,17,75)
bbox_0=Bbox(curtain,93,20,74,320,165,10,96)
bbox_1=Bbox(plants,144,46,53,320,27,27,74)
bbox_2=Bbox(air_conditioner,147,58,55,412,15,15,81)
bbox_3=Bbox(plants,58,61,52,480,24,31,70)
bbox_4=Bbox(tv_cabinet,55,103,41,480,111,19,34)
bbox_5=Bbox(tv,50,103,69,480,77,6,44)
bbox_6=Bbox(combination_sofa,119,110,45,480,166,105,48)
bbox_7=Bbox(painting,152,110,80,480,102,6,41)
bbox_8=Bbox(floor-standing_lamp,145,163,55,320,21,21,80)
bbox_9=Bbox(dining_chair,50,200,43,320,26,28,41)
bbox_10=Bbox(dining_chair,67,214,43,480,26,28,41)
bbox_11=Bbox(dining_table,49,228,50,480,77,40,43)
bbox_12=Bbox(dining_chair,67,233,43,480,26,28,41)
bbox_13=Bbox(painting,21,236,77,480,58,6,39)
bbox_14=Bbox(shoe_cabinet,113,246,68,480,52,16,120)
bbox_15=Bbox(dining_chair,41,252,43,480,26,28,41)
bbox_16=Bbox(dining_chair,67,252,43,480,26,28,41)
bbox_17=Bbox(sideboard,54,264,68,320,98,16,120)
```

Figure 12: An example conversation of SpatialLM.

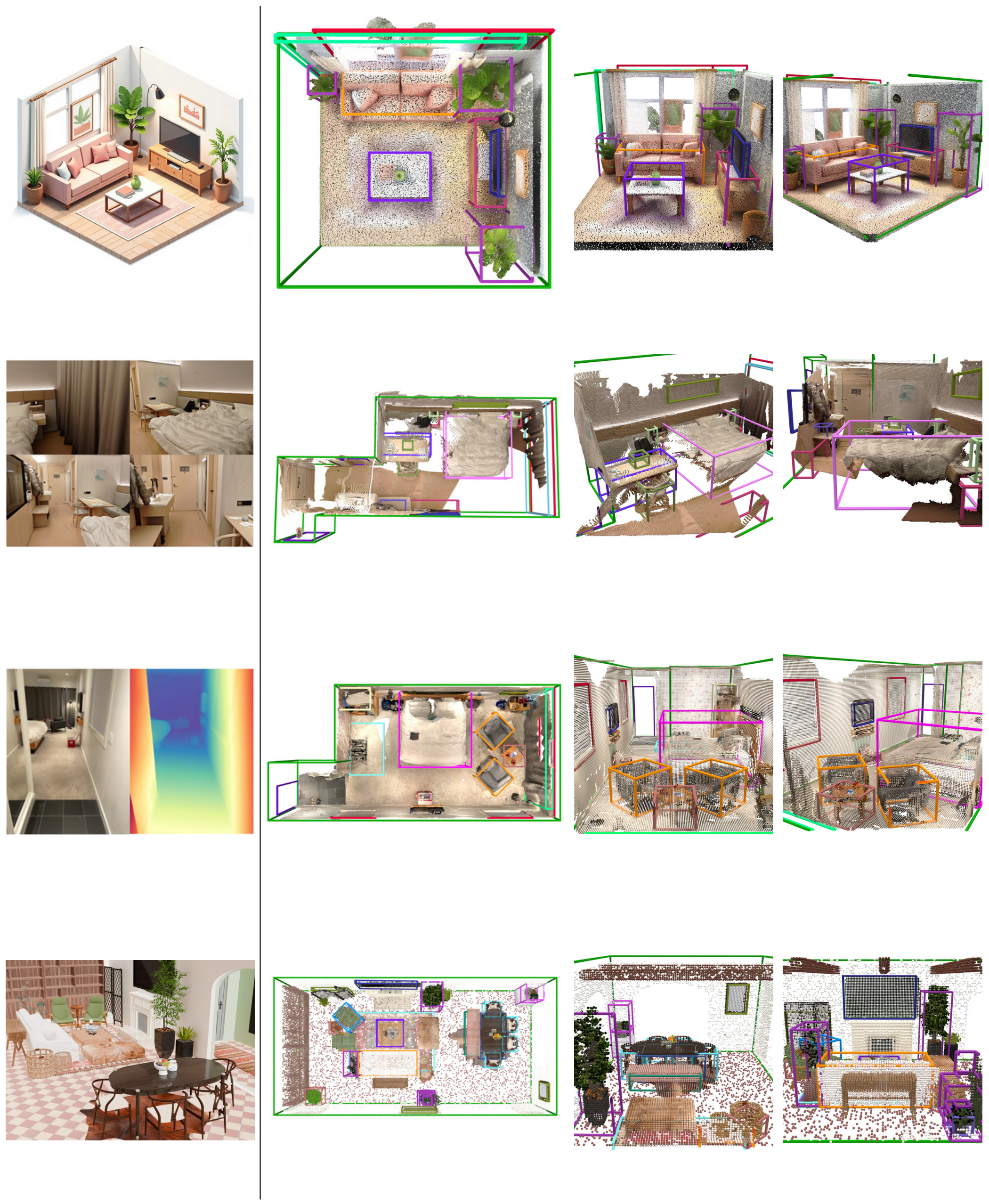


Figure 13: Qualitative results on various input sources. **First row**: Text-to-3D. **Second row**: Hand-held video camera. **Third row**: LiDAR-based reconstruction. **Fourth row**: Synthetic mesh.

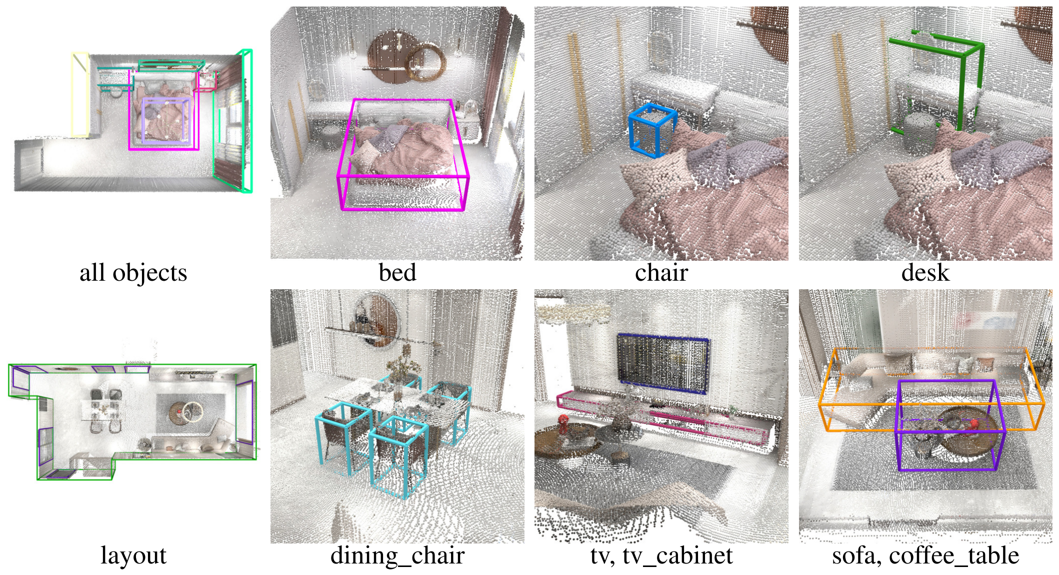


Figure 14: Qualitative results of detection with user-specified categories. Labels below each figure indicate the category specified for detection.

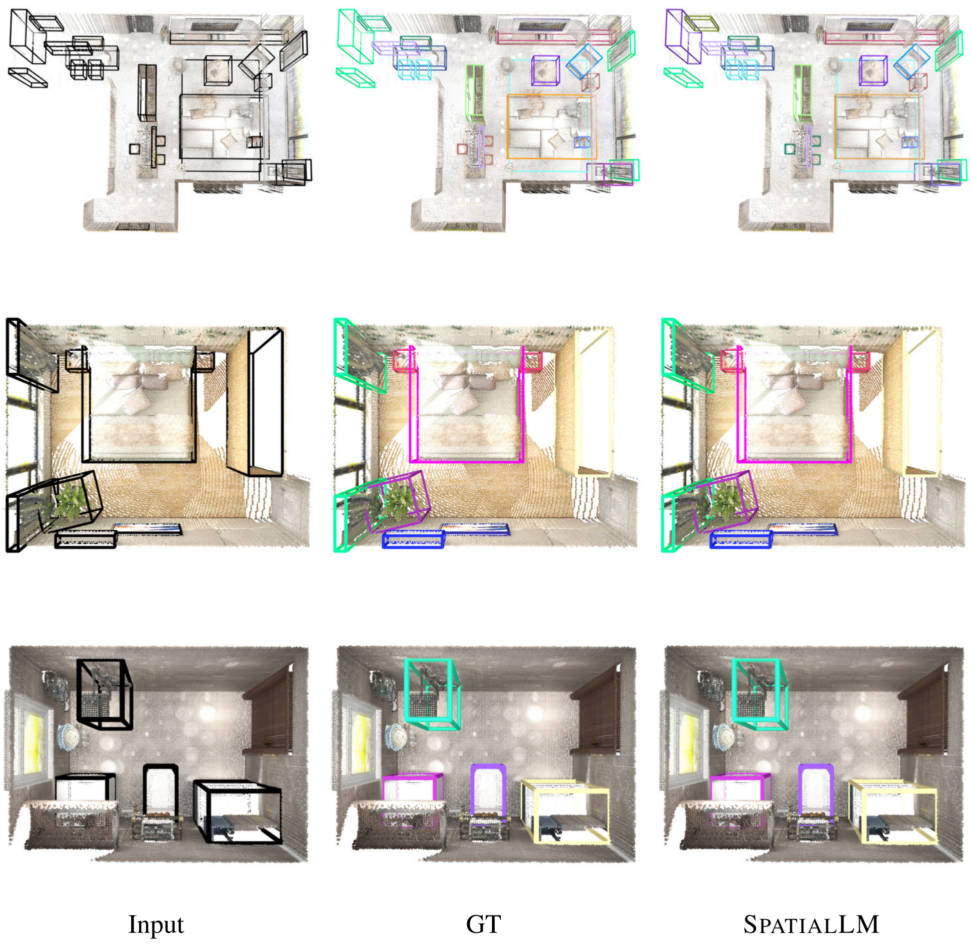


Figure 15: Qualitative results of semantic label completion. Black-colored input boxes denote objects with unknown categories, which are classified by SPATIALLM.