
Supplementary Material for SPATIALLM: Training Large Language Models for Structured Indoor Modeling

Anonymous Author(s)

Affiliation

Address

email

1 In this document, we provide more details of SPATIALLM dataset in Section 1, implementation
2 details of SPATIALLM in Section 2, supplementary results for SPATIALLM experiments in Section 3,
3 and extensions of SPATIALLM in Section 4.

4 1 SPATIALLM Dataset

5 SPATIALLM dataset comprises 12,328 distinct scenes with 54,778 rooms, featuring a total of 403,291
6 walls, 123,301 doors, and 48,887 windows, with an overall floor area of approximately 863,986 m².

7 **Object categories.** We have curated a set of 59 commonly occurring object categories, organized by
8 functional and semantic similarity, as shown in Table 1. Statistics of these categories can be found in
9 Figure 1.

Table 1: 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

10 Figure 2 plots the object-to-room correlation heatmap, which reveals strong correlations between
11 object distribution and specific room types. Additionally, each room type contains a variety of object
12 categories, highlighting the diversity of SPATIALLM dataset.

13 **Dataset visualizations.** We provide visualizations of the ground-truth point cloud and layout
14 annotations in Figure 3 to showcase the quality of the photo-realistic scans and the realism of the

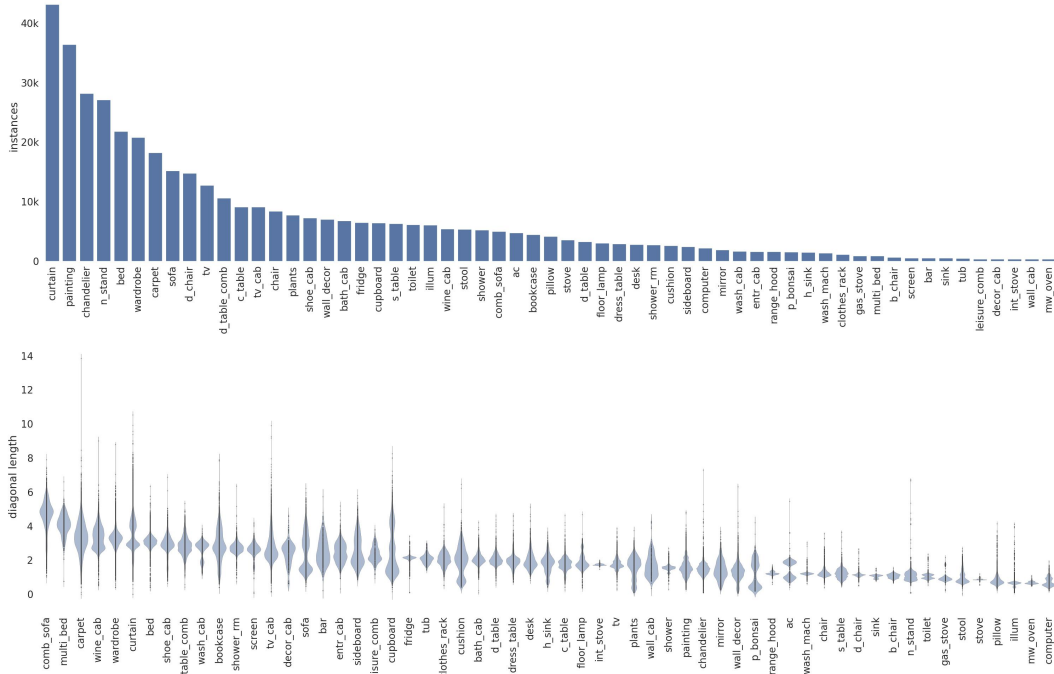


Figure 1: Statistics of the SPATIALLM dataset. **Top:** Number of objects in each category. **Bottom:** Distribution of physical sizes in each category. We compute the object size as the diagonal length of the object’s bounding box (in meters). All objects in our dataset have realistic physical sizes.

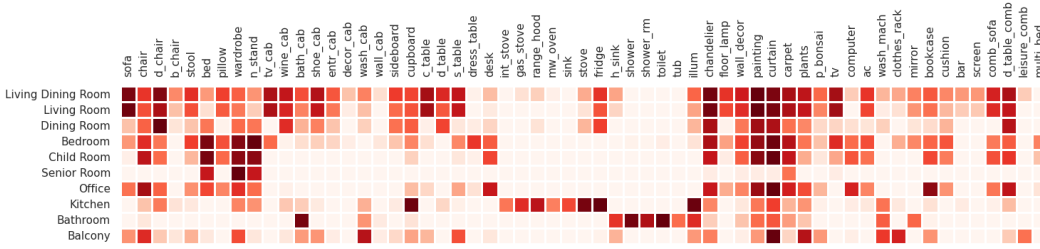


Figure 2: Room object correlation of SPATIALLM dataset.

15 layout. Additionally, we show rendered images of objects in Figure 4 to illustrate the diversity and
 16 quality of the objects in our dataset.

17 2 Implementation Details of SPATIALLM

18 In Figure 5, we show the full prompt and an example response of SPATIALLM. Note that we shift the
 19 point cloud and the corresponding layout and object box coordinates to ensure that they are all non-
 20 negative. Next, we quantize the coordinates into 1,280 bins, each with a resolution of 2.5cm. LLM
 21 predicts integer values, which are mapped back to continuous values using the inverse transformation
 22 and normalization, restoring them to the original coordinate system.

23 **Data augmentations.** We apply a number of augmentations to the point cloud, which are summarized
 24 in Table 2 and explained below.

25 We begin with a cuboid crop following V-DETR [7]. This process involves randomly selecting a
 26 point as the center of the cuboid and then determining random dimensions for the cuboid relative to
 27 the input point cloud size. We maintain a minimum aspect ratio of 0.8 and ensure that at least 50000
 28 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.

Table 2: Point cloud data augmentations.

Method	Parameters
cuboid crop	min_points:50000, aspect:0.8, min_crop:0.75, max_crop:1.0, p:0.1
random jitter	sigma:0.025, clip:0.05, ratio:0.8, p:0.9
random jitter	sigma:0.2, clip:0.2, ratio:0.05, p:0.8
random jitter	sigma:0.4, clip:1.0, ratio:0.001, p:0.75
random jitter	sigma: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

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 [10] and ScanNet [2], 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.

3 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. For each experiment, we also provide a link to an HTML page with qualitative results in the supplementary material folder.

3.1 Layout Estimation

Per-category results. In Table 3, 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.

Link to qualitative results: [visualization_layout.html](#)

Table 3: Experiment results on layout estimation.

Method	$\text{IoU}_{2D}@0.25$				$\text{IoU}_{2D}@0.5$			
	avg.	wall	door	window	avg.	wall	door	window
RoomFormer [9]	70.4	86.0	69.9	55.2	67.2	84.6	66.4	50.7
SceneScript [1]	83.1	92.9	81.6	74.7	80.8	91.9	79.7	70.9
SPATIALLM (ft. Structured3D)	32.8	54.1	25.0	19.4	17.9	33.5	12.0	8.3
SPATIALLM (ft. Ours)	50.4	72.4	37.8	41.0	36.0	64.2	28.6	15.2
SPATIALLM (ft. Ours \rightarrow Structured3D)	84.2	93.5	79.8	79.2	81.5	91.5	77.0	75.9

3.2 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 4 and Table 5, respectively.

For $\text{IoU}_{3D}@0.25$, we have +12.4% overall gain compared to SceneScript, and -3.5% difference compared to V-DETR. For $\text{IoU}_{3D}@0.5$, we have +10.5% overall gain compared to SceneScript, and -9.5% 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.

Link to qualitative results: [visualization_object.html](#)

Table 4: 3D object detection results with $\text{IoU}_{3D}@0.25$. †: trained on ScanNet only. *: trained on our dataset only. “n/a” indicates the category is not included in our dataset.

Method	average	bathub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	garbin.	picture	refridge.	s.curtain	sink	sofa	table	toilet	window
V-DETR [7]	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 [1]	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†	3.0	5.1	9.5	2.8	1.3	5.4	0.0	3.3	4.2	1.4	0.6	0.3	2.3	2.9	0.0	0.4	4.9	5.0	5.1
SPATIALLM*	29.0	31.3	55.9	28.0	12.6	31.0	n/a	15.6	31.8	n/a	n/a	9.1	24.3	n/a	7.0	38.3	27.2	64.7	n/a
SPATIALLM	61.6	83.9	78.8	59.1	36.2	81.6	50.0	59.4	65.6	59.8	49.1	21.0	49.8	59.5	65.7	76.1	60.8	86.5	65.2

Table 5: 3D object detection results with $\text{IoU}_{3D}@0.5$. †: trained on ScanNet only. *: trained on our dataset only. “n/a” indicates the category is not included in our dataset.

Method	average	bathub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	garbin.	picture	refridge.	s.curtain	sink	sofa	table	toilet	window
V-DETR [7]	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 [1]	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†	0.9	5.1	5.0	1.4	0.1	0.7	0.0	0.6	0.8	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	1.0	0.8
SPATIALLM*	17.9	21.9	46.3	13.9	5.1	15.7	n/a	4.4	17.1	n/a	n/a	2.7	19.1	n/a	1.2	25.9	13.5	46.0	n/a
SPATIALLM	47.3	64.5	70.9	46.1	26.9	71.0	31.0	42.9	57.8	38.7	35.5	8.4	36.5	38.1	41.8	62.1	53.3	82.7	42.8

3.3 Zero-shot Detection on Videos

Per-category results. We collect 107 virtual room-tour videos and use MAST3R-SLAM [6] 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 6, 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.

Link to qualitative results: [visualization_zeroshot.html](#)

Table 6: Zero-shot results on layout estimation and 3D object detection with IoU@0.25.

(a) Layout estimation

avg.	wall	door	window
52.6	66.0	52.5	39.5

(b) 3D Object detection

average	curtain	nighstand	chandelier	wardrobe	bed	sofa	chair	cabinet	diningtab.	plants	tvcab.	coffeetab.	sidetab.	aircond.	dresser	refrige.	painting	carpet	tv
33.7	32.5	69.2	38.9	35.9	95.2	68.2	24.1	2.4	17.9	29.3	26.7	61.1	10.6	15.4	33.3	8.3	29.3	28.9	14.0

4 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.

4.1 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 [4] with prompt “a low poly 3D living room in cartoon style”, then convert the image into 3D using TRELLIS [8].
- **Hand-held video camera:** We take a real-world video using a hand-held camera and perform 3D reconstruction using MAST3R-SLAM [6].
- **LiDAR-based reconstruction:** We take point clouds reconstructed from captures using iPhone ARKit from MultiScan [5].
- **Synthetic mesh:** We create point clouds by randomly sampling mesh surfaces from a synthetic dataset HSSD [3].

As can be seen in Figure 6, our model is robust to variations in point cloud origin, structure, and appearance.

4.2 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 7.

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 8. Our model achieves an overall classification accuracy of 96.8% on the test set of our dataset.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [5] 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.
- [6] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MAST3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *CVPR*, 2025.
- [7] 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.
- [8] 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. *CoRR*, abs/2412.01506, 2024.
- [9] Yuanwen Yue, Theodora Kontogianni, Konrad Schindler, and Francis Engelmann. Connecting the dots: Floorplan reconstruction using two-level queries. In *CVPR*, pages 845–854, 2023.
- [10] 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.



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



Figure 4: Example objects in SPATIALLM dataset. Note that each category has a diverse set of objects with high quality in both geometry and appearance.

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 5: An example conversation of SPATIALLM.

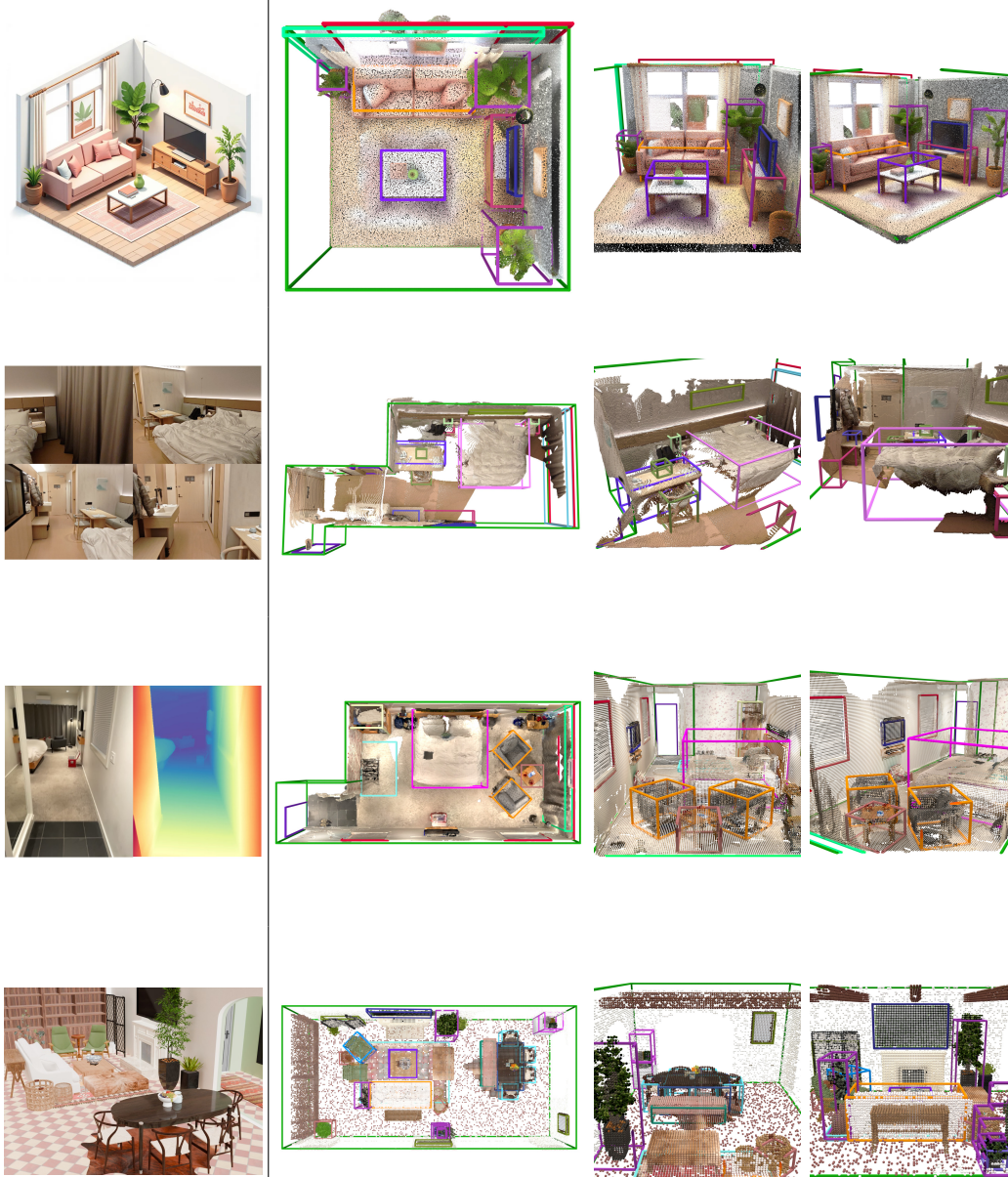


Figure 6: Qualitative results on various input sources. **First row:** Text-to-3D. **Second row:** Hand-held video camera. **Third row:** LiDAR-based reconstruction. **Fourth row:** Synthetic mesh.

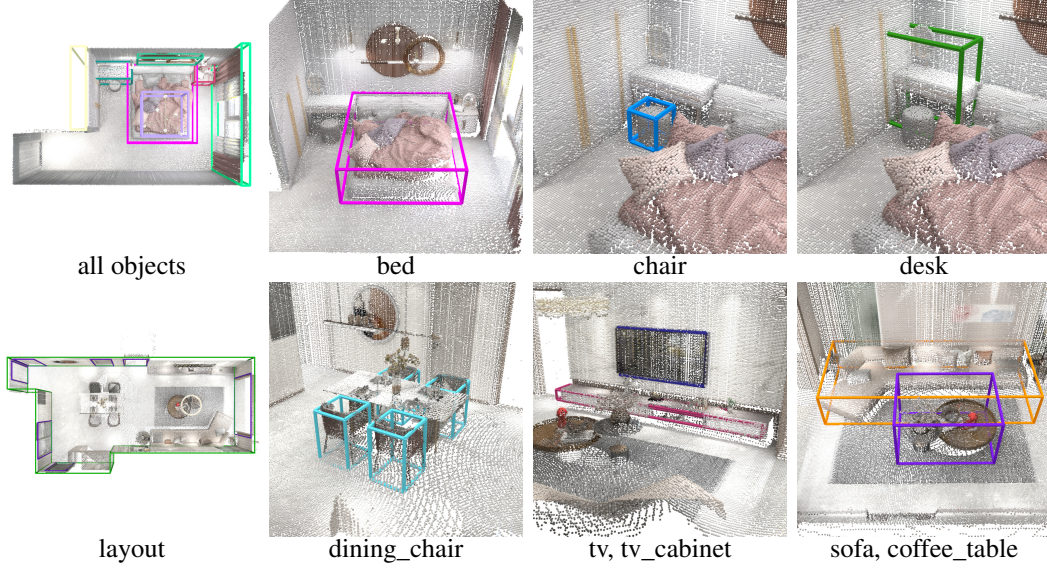


Figure 7: Qualitative results of detection with user-specified categories. Labels below each figure indicate the category specified for detection.

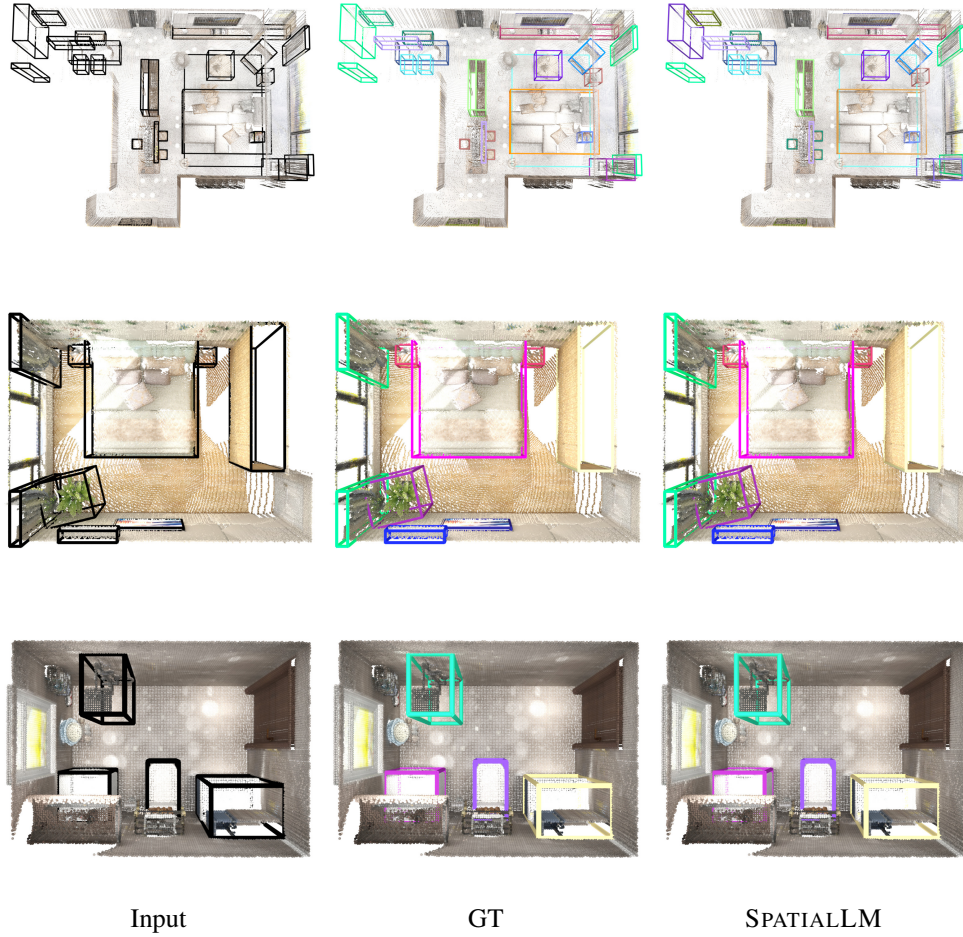


Figure 8: Qualitative results of semantic label completion. Black-colored input boxes denote objects with unknown categories, which are classified by SPATIALLM.