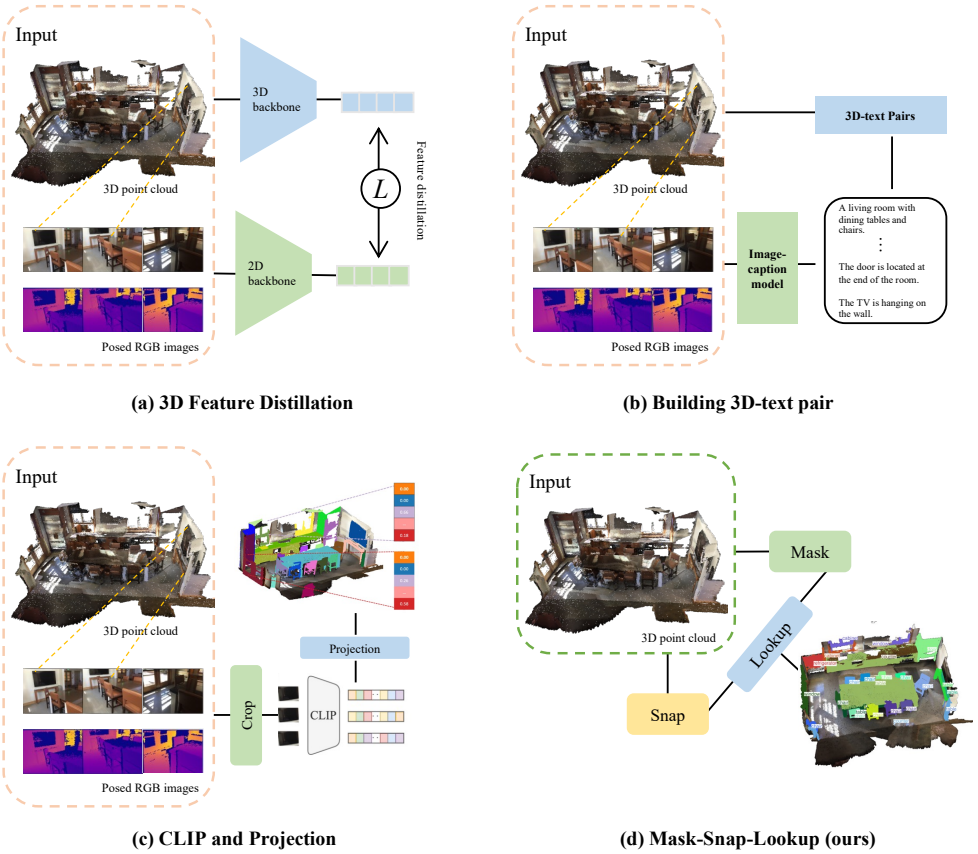
OPENINS3D: SNAP AND LOOKUP FOR 3D OPEN-VOCABULARY INSTANCE SEGMENTATION

Anonymous authors
Paper under double-blind review

APPENDIX

OpenIns3D is a powerful, 2D input-free, fast-evolving, complex-input-handling framework for 3D open-world scene understanding. We showcase the differences between OpenIns3D and other frameworks in Figure 1.



(a) 3D Feature Distillation **(b) Building 3D-text pair**

(c) CLIP and Projection **(d) Mask-Snap-Lookup (ours)**

Figure 1: **Comparison of the OpenIns3D framework with other models.** OpenIns3D offers a unique capability of using only 3D input, making it more applicable in real-life scenarios. a) 3D feature distillation frameworks, where 2D images are used as a bridge to distill language-aligned features into 3D, with typical works including OpenScene (**?**) and Clip2Scene (**?**). b) Building 3D-text pairs, where 2D captioning models are used to build 3D-text pairs for feature learning, with typical works including PLA-family (**???**). c) CLIP and Projection, where objects are cropped out of 2D images before being processed by CLIP, and the results are directly projected into 3D, including OpenMask3D (**?**), OV-3DET (**?**) and CLIP^2 (**?**). d) Mask-Snap-lookup, where only 3D input is needed for 3d open world scene understanding tasks

A MORE DETAILS ON METHODOLOGIES

A.1 CLASS-AGNOSTIC MASK PROPOSAL MODULE

We modified components in Mask3D **?** that require classification labels to make it a class-agnostic setting. This includes removing semantic probability components in Hungarian Matching, eliminating semantic classification loss, discarding classification logits-based ranking, and getting rid of classification logits-based filtering. Instead, we added the *Mask Scoring* module and *Mask Filtering* techniques to acquire high-quality mask proposals without relying on semantic labels.

Mask scoring firstly utilizing the Hungarian Match to pair N proposed masks with n ground truth masks and calculating the IoU values. For $N - n$ unmatched masks, the IoU is set to zero. This yields GT IoUs for all proposed masks. Training involves using a two-layer MLP to process N mask queries and predict their IoU values. The training is supervised by the the difference between predicted IoU and Ground Truth IoU, as shown in formula **??**.

A.2 CAMERA POSE GENERATION WITH *Lookat* FUNCTION

Here we detail how the pose matrix $Pose$ can be obtained using the *Lookat* function, followed by **?**.

Given the camera position coordinates P_{cam}, which are located even at the top of the scene, and the camera target coordinate P_{target}, which is always the centre of the scene, along with the up axis of the scene U (*i.e.* $[0, 0, -1]$), the pose matrix $Pose$ can be obtained as follows:

$$Pose = \begin{bmatrix} right_x & up_x & -forward_x & T_x \\ right_y & up_y & -forward_y & T_y \\ right_z & up_z & -forward_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Following the convention, "right" corresponds to the positive x-axis, "up" corresponds to the positive y-axis, and "forward" corresponds to the negative z-axis.

The normalized forward vector is the negative normalized direction from P_{cam} to P_{target}:

$$forward = \frac{P_{\text{cam}} - P_{\text{target}}}{\|P_{\text{cam}} - P_{\text{target}}\|}$$

The normalized right vector is the *cross-product* between the up axis U and the forward vector:

$$right = \frac{U \times forward}{\|U \times forward\|}$$

The normalized up vector is the *cross-product* between the forward vector and the right vector:

$$up = \frac{forward \times right}{\|forward \times right\|}$$

The translation values T_x, T_y, and T_z are simply the components of the camera position P_{cam}:

$$T_x = P_{\text{cam}_x}, \quad T_y = P_{\text{cam}_y}, \quad T_z = P_{\text{cam}_z}$$

Finally, the $Pose$ matrix can be obtained by assembling these values into the 4×4 matrix format.

A.3 CAMERA INTRINSIC CALIBRATION

Once the $Pose$ matrix is obtained, we initialize the intrinsic matrix with a standard intrinsic matrix $Intri$. Using both the $Pose$ and $Intri$, we perform a rapid point projection with the completed camera model, resulting in randomly positioned 2D scene. Subsequently, we uniformly scale the values of fx, fy, cx, and cy in the initialized intrinsic matrix $Intri$ by the same factor to reposition and rescale the projected image into the center of the image plane. For example, if the original projected point was located in image coordinates within the range of $[-1000, -192]$ in x, our calibrated intrinsic metrics transform it to $[0, 2000]$ in x. Crucially, we preserve the ratio between x and y coordinates to achieve the final image without any additional loss in proportion. This procedure ensures that the utilization of each image is extensive and encompasses all the proposed masks.

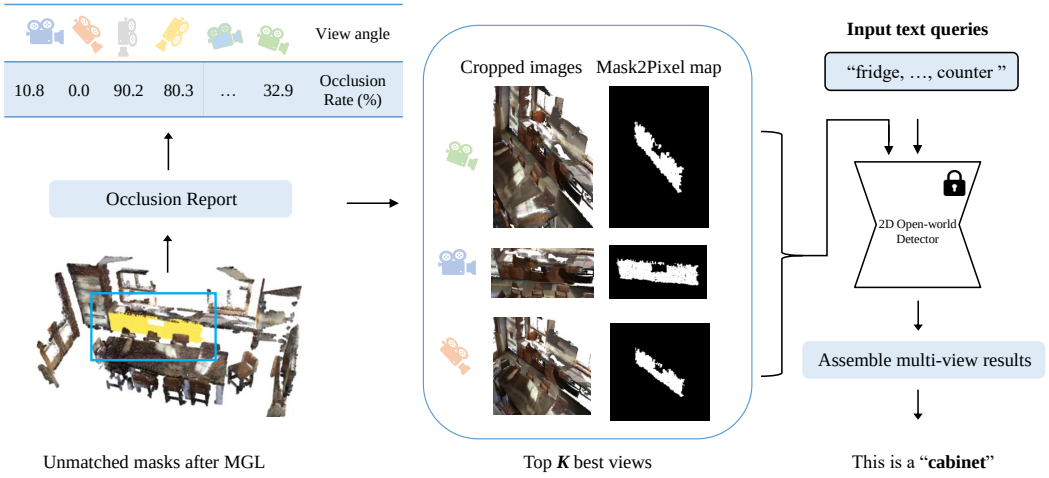


Figure 2: **Illustration of Local Enforced Lookup.** The remaining masks from phase one first go through the *Occlusion Report* module to select the best K views. The selected images are cropped before being processed by the 2D detectors to encourage a classification result.

A.4 Local Enforced Lookup

Here, we provide a detailed explanation of the *Occlusion Report* module that we proposed to effectively evaluate the occlusion condition of masks in all synthetic images. Specifically, the following four steps are executed:

- **Step 1. Point Count Array:** We initiate the process by constructing a 3D array with dimensions $W \times H \times (M+1)$, where M represents the number of masks, and 1 is added to account for the background points. This array will be denoted as PC, *i.e.* point count, as it is designed to store the number of points of the 3D mask projected onto each pixel in the images. For example, if the pixel at coordinates i, j is occupied by two points from the 3D mask k during the projection, $PT_{i,j,k}$ will be assigned the value 2.

- **Step 2. Foremost Point Identification:** Utilizing the depth map generated during the projection process, we construct a 2D array named FP with dimensions $W \times H$, which is used to identify the foremost point in each pixel and indicate the originating mask number. For example, if pixel i,j's foremost point is projected from Mask k, we denote $FP_{i,j} = k$.

- **Step 3. Occlusion Rate Calculation:** To evaluate the occlusion rate (OR) for mask k within specific images, we compute the following formula:

$$OR_k = \frac{\sum_{i=1}^{W} \sum_{j=1}^{H} PC_{i,j,k} \cdot (FP_{i,j} = k)}{T_k}$$

  where T represent the total number of point in mask k.

- **Step 4. All Images Report:** Finally, we repeat steps 1-3 for all images to obtain an overall report of the occlusion rate of each mask across all images, forming the final *Occlusion Report*.

After selecting the best view, synthetic scene-level images are cropped to focus on a specific mask proposal and then reprocessed by 2D detectors. The results are also searched with the help of Mask2Pixel maps to form the final classification prediction for the mask, as shown in Figure 2.

B IMPLEMENTATION DETAILS

B.1 MASK

The Mask Proposal Module is built upon a lightweight version of Mask3D **?** with 3 decoder layers. During training, we used 100 non-parameter queries for mask proposals. The bipartite matching process relies solely on the focal loss (weighted by 5) and dice loss (weighted by 2), without incorporating classification results. Despite not utilizing classification information, the Mask Proposal module is still capable of generating high-quality results. This is attributed to the diverse spatial distribution of 3D points, where two losses based on spatial information alone are sufficient for effective matching. For the mask quality scoring module, we set λ to 0.1 to down-weight zero IOU masks.

The Mask Proposal module is trained using the ADAM optimizer with a learning rate of 0.0003, and the one-cycle scheduler is applied. For ScanNetv2, we follow the downsampling approach described in **?**, voxelizing the original input with a resolution of 0.02. We apply a series of augmentations, including flipping, elastic distortion, random rotation, chromatic auto-contrast, chromatic jitter, and translation. For S3DIS, we adopt the same settings as described in **?**, training the MPM on Areas 1, 2, 3, 4, and 6, and testing on Area 5. For STPLS3D, we split the scene into 50-meter spans and use preprocessing steps as outlined in **?**. We follow the training and validation split on STPLS3D and evaluate the performance of the validation set. All datasets are trained for 600 epochs on a single Nvidia A100 80G GPU.

B.2 SNAP

We captured 16 images of the scene, evenly distributed along its outer boundary and focused on the centre of the scene. For all three datasets, we capture images with dimensions of 1000 x 1000 for a great trade-off between speed and performance. Additionally, to avoid the occlusion effect caused by the ceiling, we discard the top 0.3m points in the STPLS3D and ScanNetv2 datasets. As a result, the ceiling categories in the S3DIS dataset are completely discarded. We assign it a value of 0 in all matrices when evaluating and comparing with other methods. For STPLS3D, the camera position is located 5m higher than the top of the scene to acquire a better view.

B.3 LOOKUP

During the *Lookup* stage, we only assign a classification label to each mask if the results have been verified in at least two views. This approach ensures a higher level of confidence in the assigned class labels. In the case of *Local Enforced Lookup*, we crop the images using bounding boxes that are twice the size of the target masks. The results are then fed into 2D detectors to refine the results. Mask2Pixel maps, in this case, binary maps, are used to accurately search for the detection results, as shown in Figure 2.

C EXPERIMENT ON SCANNET200

We further evaluate OpenIns3D's performance on a more challenging dataset ScanNet200 (**?**), which features a larger vocabulary and more categories. ScanNet200 comprises 200 classes, and based on the frequency of labeled points in the training set, these 200 classes are split into three groups: "head," which contains 66 categories; "common," which contains 68 categories; and "tail," which contains 66 categories. We report the results for each category groups.

While most other OpenScene models require 2D images during inference, the OpenScene 3D distillation model can process 3D point clouds directly for Open world understanding, making it also a 2d-input free model. Compared with this model, OpenIns3D demonstrates stronger performance across all category groups, with a notable 5.4% improvement, in head group.

In comparison to methods that use 2D aligned images during inference, OpenIns3D performs competitively on the head categories, yet its performance drops in the common and tail categories. This decline stems from OpenIns3D's sole reliance on the input 3D reconstruction. Within this

reconstruction, visual information for many small objects, particularly those in the common and tail groups, are likely to be diluted or lost, resulting in less optimal performance on these categories.

On the other hand, OpenMask3D leverages original 2D image for mask understanding, showcasing robust performance across all categories. However, this enhanced performance comes at the cost of incorporation of additional modality, leading to a reduction in flexibility in its application.

Table 1: **3D instance segmentation results on the ScanNet200 validation set.** OpenIns3D demonstrates robust performance, when compared to 2D-input free models. In comparison with models utilizing 2D images, it maintains competitive performance within the head categories split. However, notable limitations emerge when dealing with small objects in the common and tail classes.

Model	Image Features	use 2D	head (AP)	common (AP)	tail (AP)	AP	AP_{50}	AP_{25}
OpenScene (2D Fusion) + masks	OpenSeg	✓	13.4	11.6	9.9	11.7	15.2	17.8
OpenScene (2D/3D Ens.) + masks	OpenSeg	✓	11.0	3.2	1.1	5.3	6.7	8.1
OpenScene (2D Fusion) + masks	LSeg	✓	14.5	2.5	1.1	6.0	7.7	8.5
OpenMask3D	CLIP	✓	17.1	14.1	14.9	15.4	19.9	23.1
OpenScene (3D Distill) + masks	OpenSeg	✗	10.6	2.6	0.7	4.8	6.2	7.2
OpenIns3D	ODISE	✗	**16.0**	**6.5**	**4.2**	**8.8**	**10.3**	**14.4**

D Per Categories Analysis

D.1 Comparison with SOTA

Table 2 and Table 3 provide the per-class results of the proposed OpenIns3D on the S3DIS and ScanNetv2 datasets. We follow the performance of PLA and highlight the novel (unseen) classes. Note per categories results for RegionPLA and Lowis3D are not available. Table 4 represents the per-categories results for STPLS3D data, compared with PointCLIP and PointCLIPV2.

Table 2: **Per-class Results of 3D Open-vocabulary Instance Segmentation on S3DIS AP50.** Performance on novel classes is marked in blue.

Methods	Partition	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board
PLA ?	B8/N4	89.5	100.0	50.8	00.0	35.3	36.2	60.5	00.1	84.6	01.9	00.8	59.4
	B6/N6	89.5	60.2	17.9	00.0	41.5	10.2	02.1	00.6	86.2	45.1	00.1	02.2
OpenIns3D	–/N12	00.0	84.4	29.0	00.0	00.0	62.6	25.2	25.5	52.0	60.0	00.0	00.0

Table 3: **Per-class Results of 3D Open-vocabulary Instance Segmentation on ScanNet AP50.** Performance on novel classes is marked in blue.

Methods	Partition	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	shower c.	toilet	sink	bathtub
PLA ?	B13/N4	50.5	77.0	82.9	43.4	75.4	49.0	46.0	43.7	46.5	33.7	23.2	54.1	49.6	56.0	97.8	47.5	85.8
	B10/N7	53.7	62.7	11.2	70.5	27.2	47.7	45.7	30.0	01.5	39.9	40.8	50.6	68.6	84.6	92.9	24.6	00.0
	B8/N9	45.1	77.4	82.2	84.2	74.2	48.9	51.0	30.0	00.5	02.1	16.8	44.9	28.3	35.1	94.3	16.6	00.0
OpenIns3D	–/N17	24.3	52.5	75.7	61.6	40.6	39.7	45.5	54.8	0.5	33.5	16.7	48.1	18.5	4.3	50.1	16.8	7.6

Table 4: **Per-class Results of 3D Open-vocabulary Instance Segmentation on STPLS3D AP50.** All models are tested in a zero-shot manner.

Methods	building	veg	vehicle	truck	aircraft	mil-veh	bike	motorbike	lightpole	signs	clutter	fence
PointCLIP	15.3	0.4	10.2	06.6	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
PointCLIPV2	20.3	0.2	12.3	5.8	00.0	00.0	00.0	00.0	00.0	00.0	00.0	00.0
OpenIns3D	**40.4**	**01.2**	**54.2**	**24.2**	**30.0**	**05.5**	**02.1**	**03.0**	00.0	00.0	00.0	**08.3**

In S3DIS, OpenIns3D consistently achieves high results for novel classes. We attribute this to the high quality of 3D point data in S3DIS, which ensures favourable conditions for recognition in Snap images.

However, for classes like columns, OpenIns3D struggles to produce meaningful results. Our performance on categories such as windows, floors, doors, tables, and sofas is typically at least 20% higher than PLA results. It is worth noting that PLA is partially trained on the base class and requires 2D images for captioning purposes.

For the ScanNetv2 dataset, our model performs better than PLA in certain categories such as chairs, sofas, tables, bookshelves, pictures, counters, and bathtubs. However, it slightly underperforms PLA in categories like beds, fridges, shower curtains, toilets, and sinks. In ScanNetv2, the quality of the point cloud data is not very high, especially for scans with higher scene IDs. As a result, the quality of Snap output is limited by the original point cloud, leading to slightly lower performance. Nevertheless, OpenIns3D, as a 2D input-free and label-free scheme, still achieves competitive performance on the ScanNetv2 dataset.

In the case of STPLS3D, our model outperforms PointCLIP and PointCLIPV2 by a significant margin in almost every category, achieving very high results, particularly in categories such as buildings, vehicles, trucks, and aircraft. However, the performance on very small objects, such as bikes, motorbikes, signs, and light poles, is not as strong. This is because the Snap module positions the camera at a high-level point to capture point cloud data from buildings, resulting in a limited number of pixels available for these smaller objects. This presents a challenge for OpenIns3D.

D.2 Cross-domain Analysis

Table 5 presents the per-category results for the cross-domain OpenIns3D model, trained on S3DIS and tested on ScanNetv2. Despite the performance being relatively lower than that of the in-domain model (trained and tested on the same dataset), the performance is still competitive when compared to other SOTA results. Note that these SOTA models use pre-trained 2D/3D models to propose bounding boxes, which are trained with in-domain data (ScanNetv2)

The class categories between S3DIS and ScanNetv2 are very different. Within the 17 classes in ScanNetv2, only 6 classes exist in S3DIS. However, OpenIns3D, trained on S3DIS, can still perform relatively well in other categories that have never been seen before, demonstrating its good generalization capability.

This outcome is as anticipated, as the design of MPM closely resembles that of SAM in the 2D context, which has shown remarkable capabilities in mask proposal generation after being trained on a substantial amount of data. We believe that with more 3D class-agnostic labels available, the MPM module is capable of generating higher-quality Class-agnostic mask proposals.

Table 5: **Cross-domain Analysis of OpenIns3D on OVOD on ScanNetv2 AP25.** OpenIns3D achieves competitive results on the cross-domain dataset, even on categories at are not available on the training dataset, highlighted in blue. Compared with other SOTA models on OVOD, cross-domain OpenIns3D still has competitive performance. MPM-SC: MPM trained on ScanNetv2; MPM-S3: MPM trained on S3DIS.

Methods	BBox Prop	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	shower c.	toilet	sink	bathtub
OpenIns3D	MPM-SC	17.1	57.5	74.5	59.2	36.9	29.3	47.5	26.4	0.0	31.1	32.2	55.4	39.1	0.0	57.4	42.1	6.6
OpenIns3D	MPM-S3	16.1	43.5	45.7	41.8	28.6	17.7	18.3	31.9	1.2	1.0	29.3	23.1	20.1	8.0	63.6	16.4	1.7
SOTA models																		
PointCLIP	P-3DE	6.0	4.8	45.2	4.8	7.4	4.6	2.2	-	-	1.0	4.0	-	-	-	-	13.4	6.5
PointCLIPV2	P-3DE	19.3	21.0	61.9	15.6	23.8	13.2	17.4	-	-	12.4	21.4	-	-	-	-	14.5	16.8
OV-3DET	P-2DE	3.0	42.3	27.1	31.5	14.2	9.6	-	5.6	-	0.3	19.7	10.5	11.0	-	57.3	31.6	56.3

E Other Attempts for Image Generation

Figures 3 and Table 6 illustrate the alternative approaches we explored before arriving at the conclusion that synthetic scene-level images offer the optimal solution for open vocabulary instance

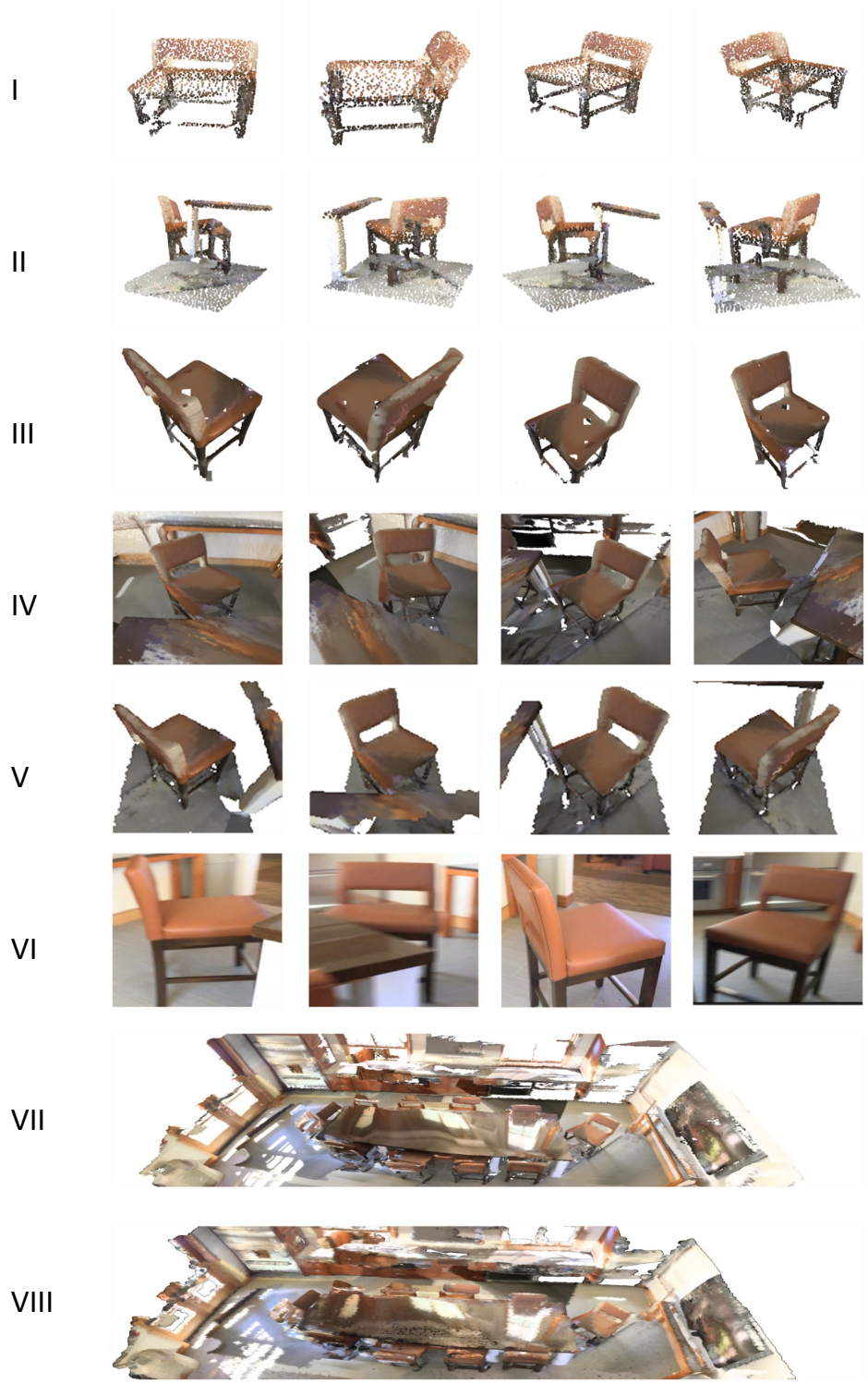


Figure 3: **Visualization of Attempts Made to Generate 2D Images from 3D.** I: LAR-point projection; II: LAR-point-bg-project; III: Mesh rendering; IV Mesh-in-scene Rendering; V: Mesh-bg-Rendering; VI: Cropped from Original 2D images; VII: Scene Level Rendering from Mesh; VIII: Scene Level Rendering from Point. Performance can be found in Table 6.

Table 6: **Evolution of *Snap* and *Lookup* Module.** The corresponding image visualization is shown in Figure 3. Scene-level rendering not only requires fewer images but also achieves superb results when compared to other pre-mask levels of rendering. *: The image sizes of VI are adjusted to fit the size of the mask area on the original images.

Idx	Methods	Job intensity	Imgs needed	Original 2D	Img size	2D backbone	AP50	AP25
I	LAR-point projection	per mask	250	✗	128^2	CLIP	5.3	8.6
II	LAR-point-bg-projection	per mask	250	✗	128^2	CLIP	6.3	10.5
III	mesh-rendering	per mask	250	✗	128^2	CLIP	6.8	7.2
IV	mesh-scene-rendering	per mask	250	✗	128^2	CLIP	6.7	7.3
V	mesh-bg-rendering	per mask	250	✗	128^2	CLIP	4.3	5.3
VI	crop-original2d	per mask	250	✓	$-^*$	CLIP	24.3	29.6
VII	scene-mesh-rendering	per scene	8	✗	1000^2	ODISE	18.8	29.8
VIII	scene-mesh-rendering	per scene	8	✗	1000^2	ODISE	28.7	38.9
IX	scene-point-rendering	per scene	8	✗	1000^2	ODISE	21.5	33.6

segmentation. These methods primarily relied on per-mask rendering, i.e. generating multiple 2D images for each mask.

Attempts I, II: Inspired by the success of LAR **?**, which uses synthetic images of objects to assist in the 3D visual grounding task, we first explore a similar approach. This involves positioning the camera around the object and projecting point clouds to generate multi-view images for each mask. However, these approaches yielded unsatisfactory performance when integrated with the CLIP model, even with their background being projected. This is mainly due to the fact that many masks are too broken, and the projected images are difficult to recognize even for humans.

Attempts III, IV, V: We redirected our attention to the mesh model. Although meshes are not always available for all 3D point clouds, we decided to investigate whether the mesh model could enhance the quality of rendered images, thereby making them more recognizable with 2D models. However, the outcomes of pre-mask rendering III, IV, V still encountered challenges in achieving reasonable performance, not to mention the considerable rendering time they demanded. The problem still boils down to the quality of the point clouds themselves. For masks with very clear and complete point clouds, such as the chair presented in Figures 3, these approaches can produce reasonable results. However, most masks are very difficult to recognize even in the mesh model. Even for human beings, a substantial amount of contextual information is required to understand those broken, distorted, and sparse mask instances.

Attempt VI: We then start to experiment with using original images and cropping out masks in the images for evaluation VI. We believe this offers the best quality of images, therefore making them most likely to be recognizable with 2D models. We use *Occlusion Reports* methods to select the top K views from all frames and crop out mask pixels with an enlarged bounding box. This approach did achieve notable performance, primarily due to the high quality of 2D images. **However, we ultimately abandoned this approach due to concerns about its applicability in general scenarios for two reasons:**

1. Such an approach requires well-aligned 2D images in both the training and inference stages. Our argument is that if well-aligned 2D images are readily available and can be seamlessly linked with 3D data, meaning they have pose, intrinsic, and depth information, it is much easier to approach open-world tasks from a 2D perspective. The open-world understanding of 3D can be achieved by conducting open-world detection in 2D images and projecting the results into 3D point clouds using available camera models and depth maps. Similar approaches have been shown to be feasible in SAM3D **?**. Therefore, the motivation for such an approach is questionable.
2. In many instances, 2D images occupy a significant portion of storage space, and it would be more practical to not rely solely on 2D images. For example, in ScanNetv2, the point cloud/mesh file of a scene occupies around 3MB, while the entire 2D image collection consumes roughly 3GB. Additionally, in many cases, 2D images are not available, as discussed in the introduction section.

Attempt VII, VIII, XI: Turning our attention to scene-level rendering, our model demonstrated a significant enhancement in performance (VII) with a notable increase of 12.0% on AP50. This

was because, by observing all broken instances from a distance and incorporating a large amount of contextual information, objects became clear and recognizable. Switching the 2D backbone from Grounding-DINO to ODISE offers further improvement (+8.9%), as ODISE proves to be more robust for diverse input queries. While rendering by the mesh model contributed to improved clarity (VIII) in images, leading to enhanced results, rendering from the point cloud made the approach more applicable (XI), and the performance remained decent when compared to other state-of-the-art methods. This is why we rendered from the point cloud in most of the datasets we tested.

F Limitations and Future Work

OpenIns3D is a novel framework that achieves remarkable performance in open-vocabulary instance segmentation, surpassing many existing methods. However, there are some limitations of OpenIns3D that need further investigation in future studies.

- Reliance on Ground Truth Instance Masks: Similar to SAM **?**, OpenIns3D still relies on ground truth mask supervision. While it does prove to have the capability to generalize masks that have never been seen before, the performance on mask proposal at scale still has a large room for improvement. Learning from the success of SAM, a query-based transformer decoder backbone could yield impressive results when trained with a large-scale dataset. There is a line of research dedicated to learning 3D class-agnostic masks, exemplified by approaches like UnScene3D **?** and SAM3D **?**. These approaches could serve as either alternatives to MPM or as a means of generating class-agnostic labels at scale for MPM to be trained on. Exploring this path further could be an interesting avenue for future research.
- Limited Performance in Semantic Segmentation: OpenIns3D heavily relies on filtering to refine the mask proposals, discarding masks with low quality directly. While this approach benefits instance segmentation by reducing false positive instances, it may limit its performance in semantic segmentation. We have also calculated the semantic segmentation results of OpenIns3D on four categories, as reported by OpenScene **?**, as shown in Table 7. Our method still exhibits a gap compared to OpenScene in terms of semantic segmentation.
- Small Object Performance: As shown in Table 1, the performance of OpenIns3D is ultimately closely linked to the quality of the point cloud itself. Masks that are very small or made of sparse point clouds would be difficult to recognize in the rendered images, as they either occupy a small portion of the image pixels or are too fragmented to be detected by the 2D models.

Nonetheless, while most researchers in the community are focused on aligning point and image features for open-world capabilities in the 3D domain, we aim to propose a simple, flexible, and powerful framework that requires no 2D input but can still achieve impressive results. OpenIns3D can easily evolve with the rapid development of 2D open-world models. In this era of rapid evolution of foundation models, we believe this attribute makes OpenIns3D powerful in many settings.

Table 7: **Comparison with OpenScene and other Frameworks on Semantic Segmentation.** Our framework prioritises mask quality and suffers overall semantic segmentation results.

Semantic Seg.	mIoU					mAcc				
Methods	Bookshelf	Desk	Sofa	Toilet	Mean	Bookshelf	Desk	Sofa	Toilet	Mean
3DGenZ (**?**)	6.3	3.3	13.1	8.1	7.7	13.4	5.9	5.9	26.3	12.9
MSeg Voting (**?**)	47.8	40.3	56.5	68.8	53.3	50.1	67.7	67.7	81.0	66.6
OpenScene-LSeg (**?**)	**67.1**	**46.4**	60.2	**77.5**	**62.8**	**85.5**	**69.5**	69.5	90.0	78.6
OpenScene-OpenSeg (**?**)	64.1	27.4	49.6	63.7	51.2	73.7	73.4	73.4	95.3	79.0
OpenIns3D	54.8	16.7	**61.6**	50.6	45.9	59.0	32.3	**76.7**	79.8	61.9

G Visualization

Mask Proposal Figure 4 and 5 present a qualitative evaluation of the mask proposal module. The learned mask proposals exhibit great similarity to the ground truth masks, often capturing additional unlabeled masks. This demonstrates the effectiveness of our class-label-free learning scheme in

producing high-quality class-agnostic mask proposals. Moreover, through the application of *Mask Scoring* and *Mask Filtering* techniques, we are able to connect fragmented or fragile masks, resulting in a substantial improvement in mask quality. These advancements provide a strong foundation for the Snap and Lookup understanding scheme.

Snap visualization Figure 6, 7 and 8 demonstrate the capability of the Snap module. With the proposed pose and intrinsic optimization scheme, the Snap module is capable of generating decent-quality images from point clouds, regardless of whether the dataset is indoor or outdoor.

Lookup Results Visualization The Lookup module effectively links 2D results with 3D. Here, we present visualizations of its outcomes from all three datasets (Figure 9, 10, 11). OpenIns3D is capable of capturing the most interesting objects in the scene without relying on any corresponding 2D images.

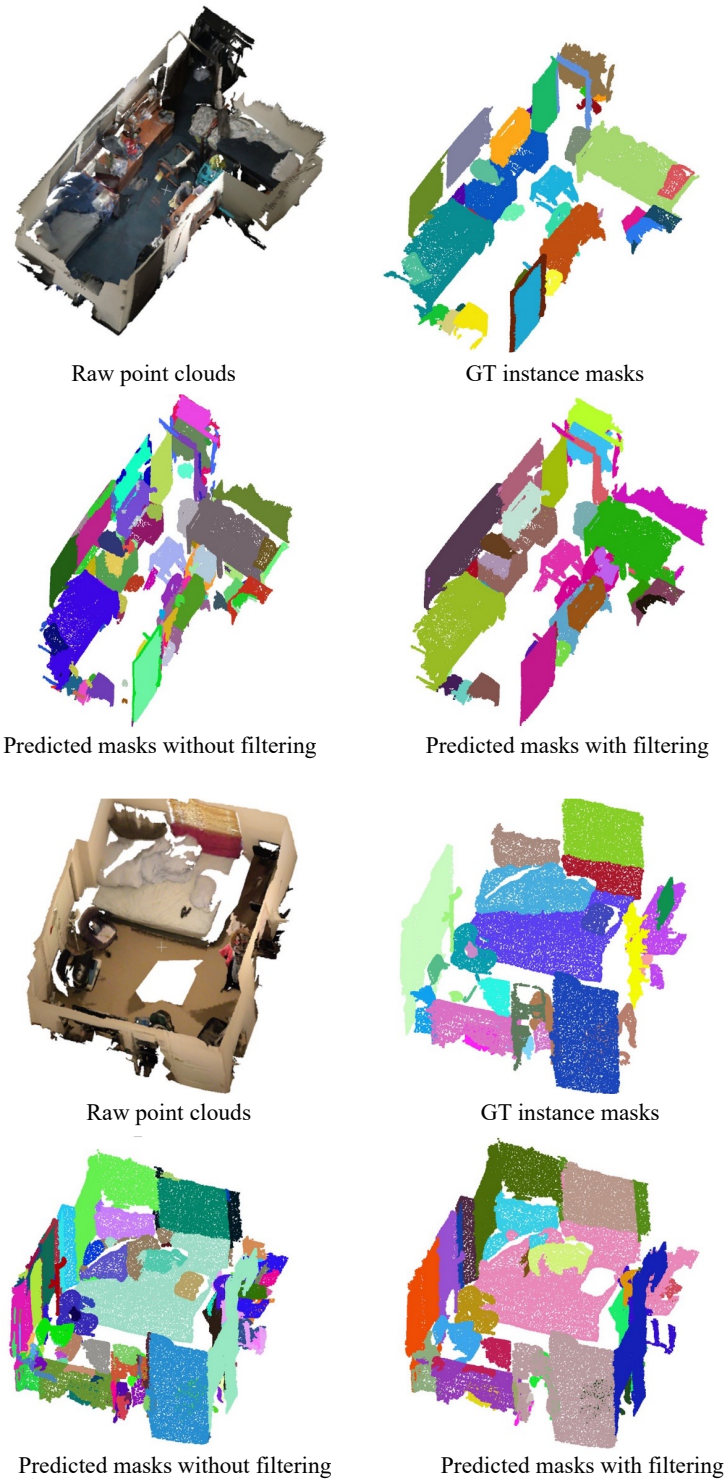


Figure 4: **Qualitative Evaluation of the Mask Proposals.** Our class-label-free approach produces high-quality masks that closely resemble the ground truth. Additionally, the incorporation of *Mask Scoring* and *Mask Filtering* further enhances the overall quality of the masks. Quantitative evaluation is shown in Table **??**.

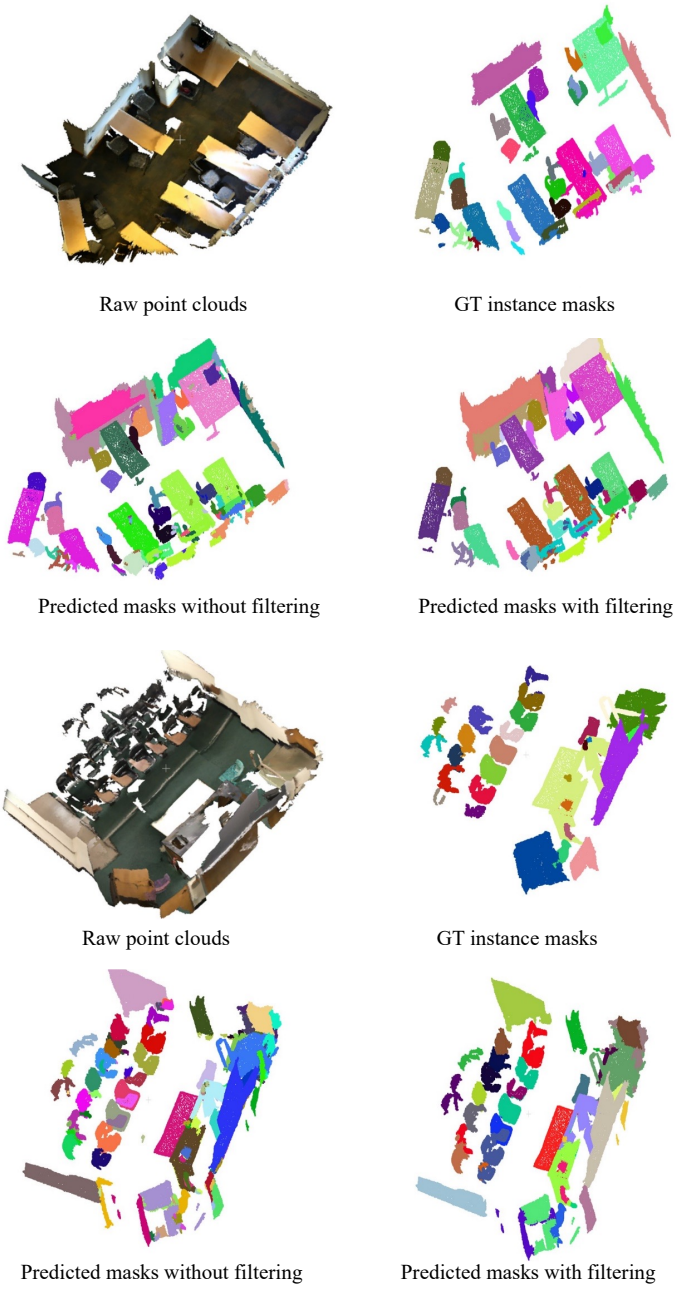


Figure 5: **Qualitative Evaluation of the Mask Proposals.** Our class-label-free approach produces high-quality masks that closely resemble the ground truth. Additionally, the incorporation of *Mask Scoring* and *Mask Filtering* further enhances the overall quality of the masks. Quantitative evaluation is shown in Table **??**.

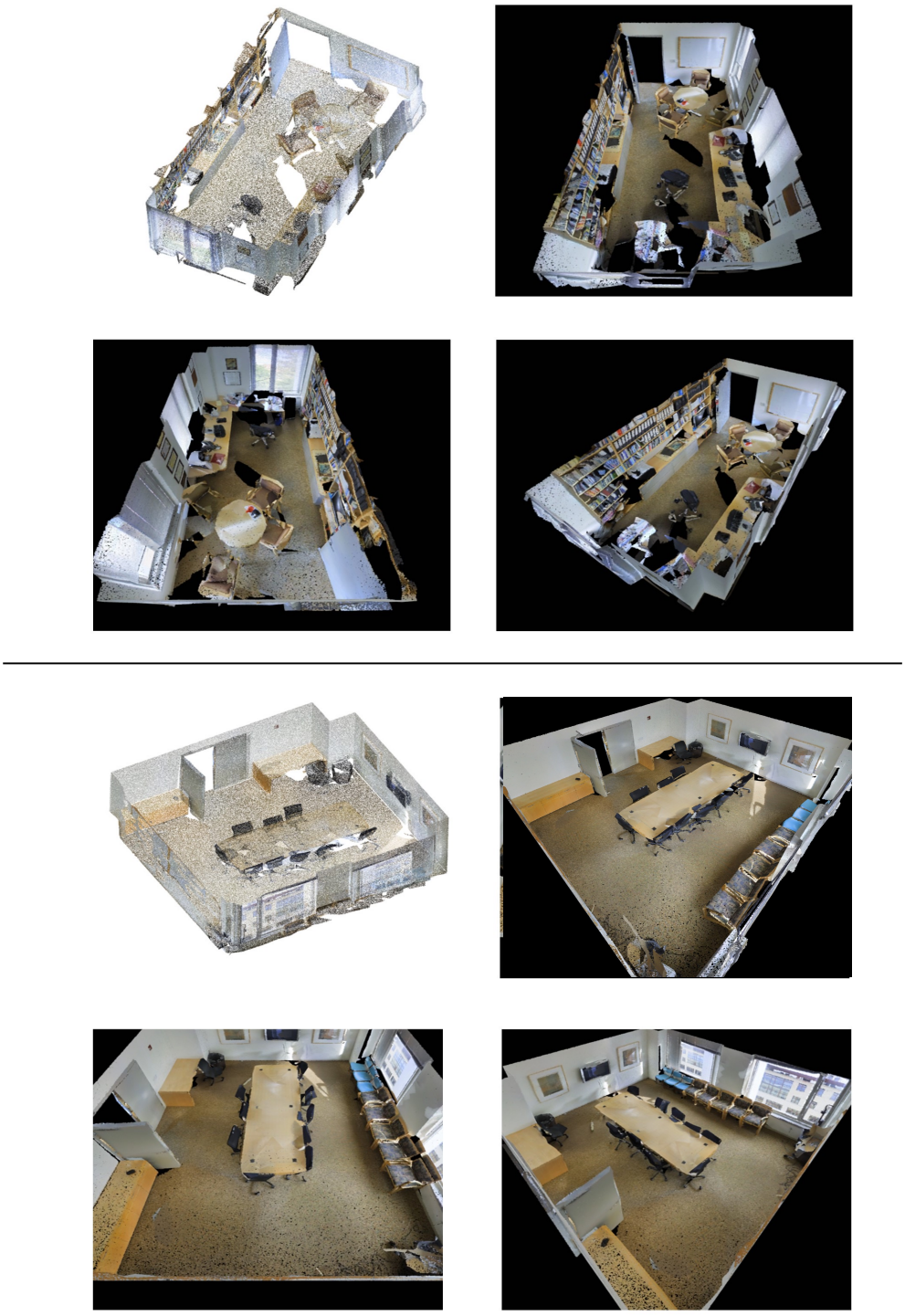


Figure 6: **Synthetic Scene-level Images of S3DIS Generated by *Snap*.** The first image is the original spare point cloud, and the following three images are outcomes of the *Snap* module.

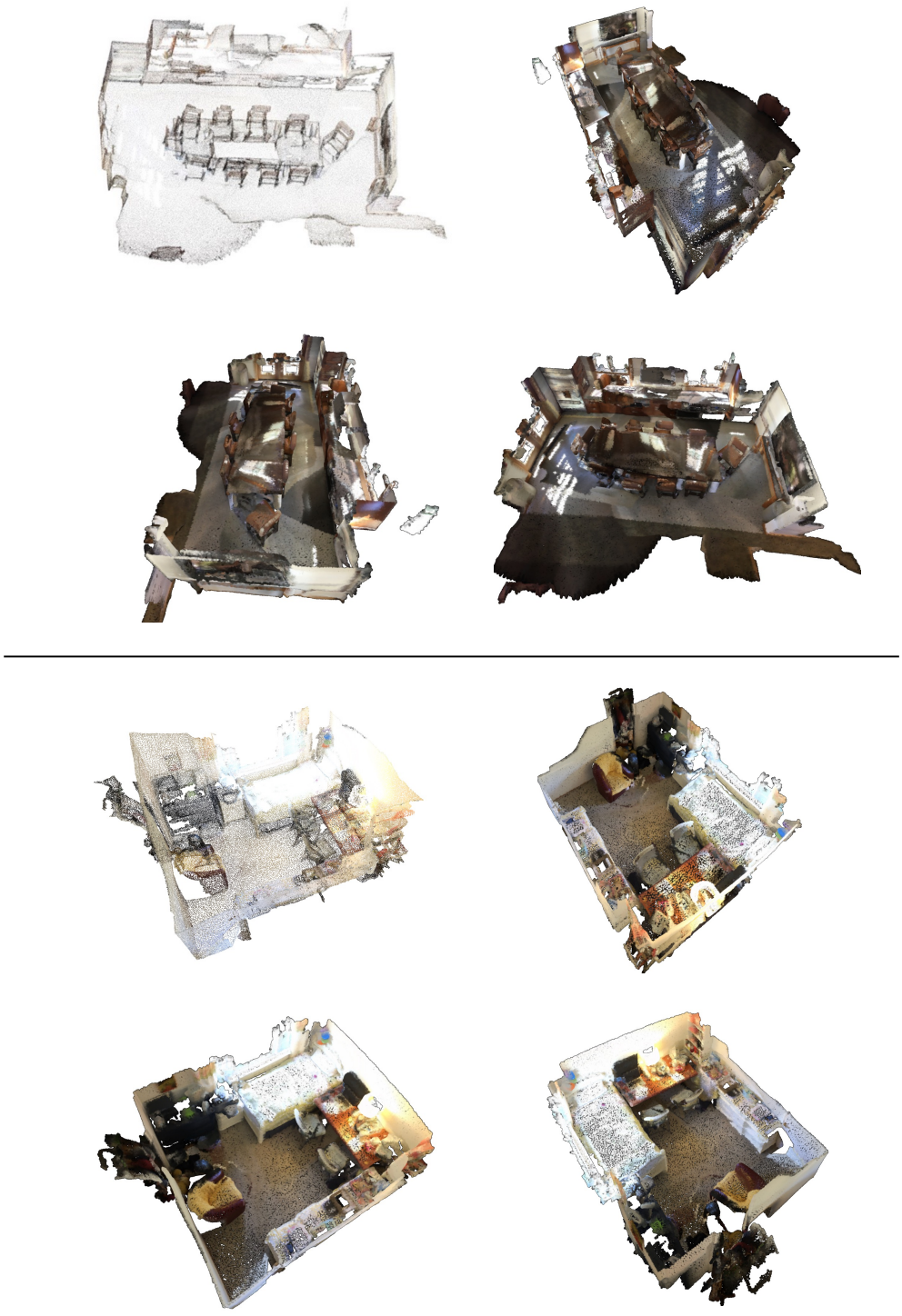


Figure 7: **Synthetic Scene-level Images of ScanNetv2 Generated by *Snap*.** The first image is the original spare point cloud, and the following three images are outcomes of the *Snap* module.

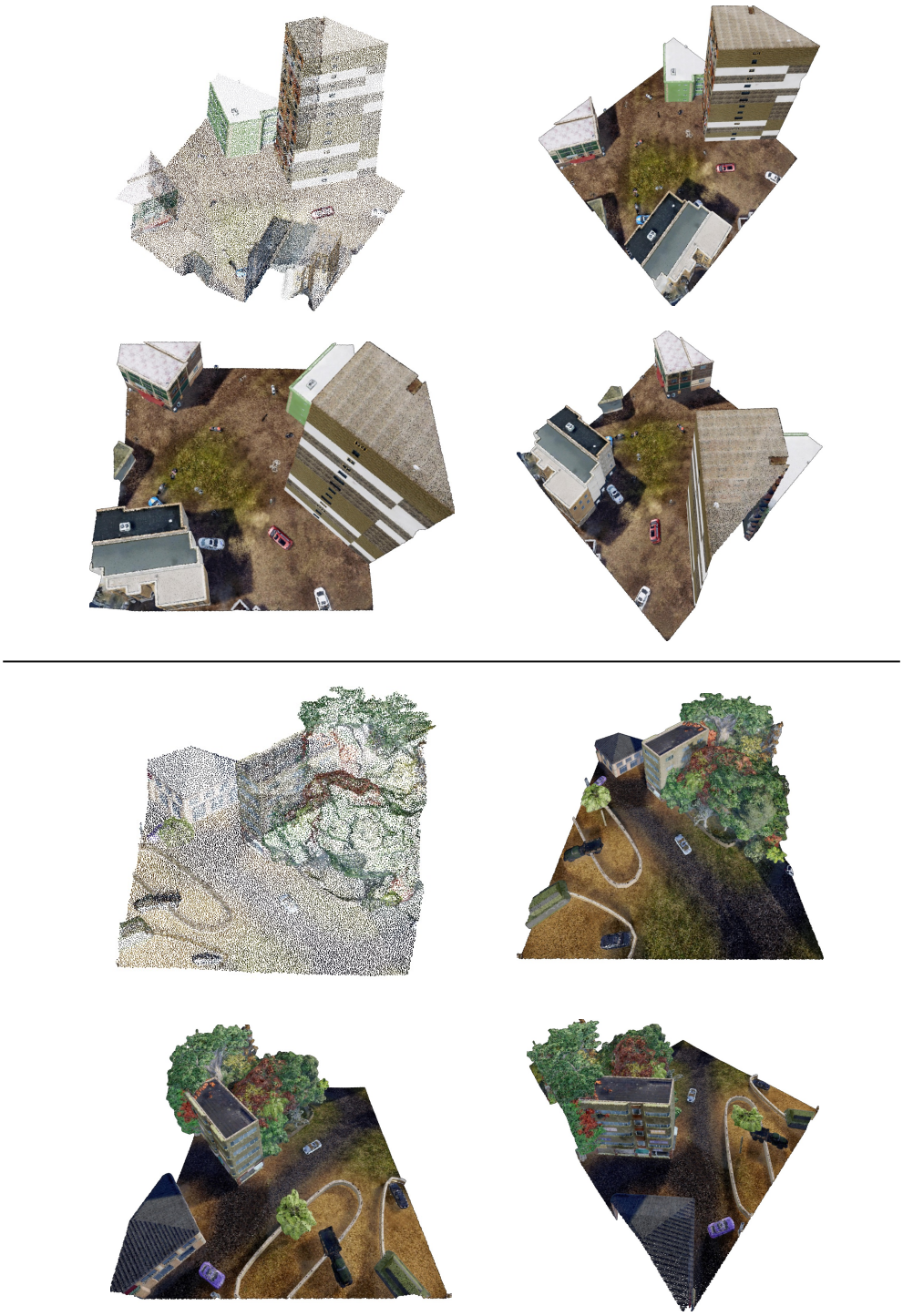


Figure 8: **Synthetic Scene-level Images of STPLS3D Generated by *Snap*.** The first image is the original spare point cloud, and the following three images are outcomes of the *Snap* module.

Input Queries: *'ceiling', 'floor', 'wall', 'beam', 'column', 'window', 'door', 'table', 'chair', 'sofa', 'bookcase', 'board'*

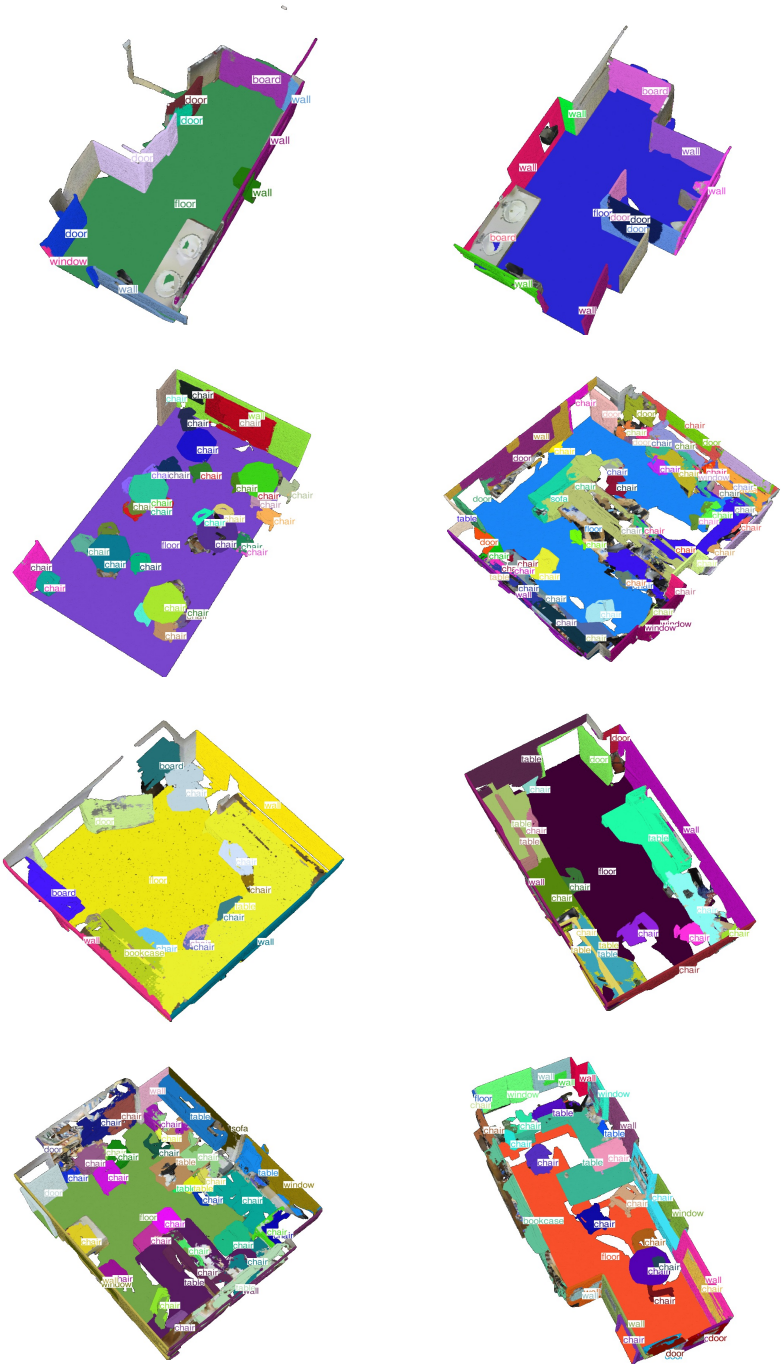


Figure 9: **Open-vocabulary Instance Segmentation Results of S3DIS by OpenIns3D (ODISE).** Instance and class labels are presented in the same color.

Input Queries: *'cabinet', 'bed', 'chair', 'sofa', 'table', 'door', 'window', 'bookshelf', 'picture', 'counter', 'desk', 'curtain', 'refrigerator', 'shower curtain', 'toilet', 'sink', 'bathtub'*

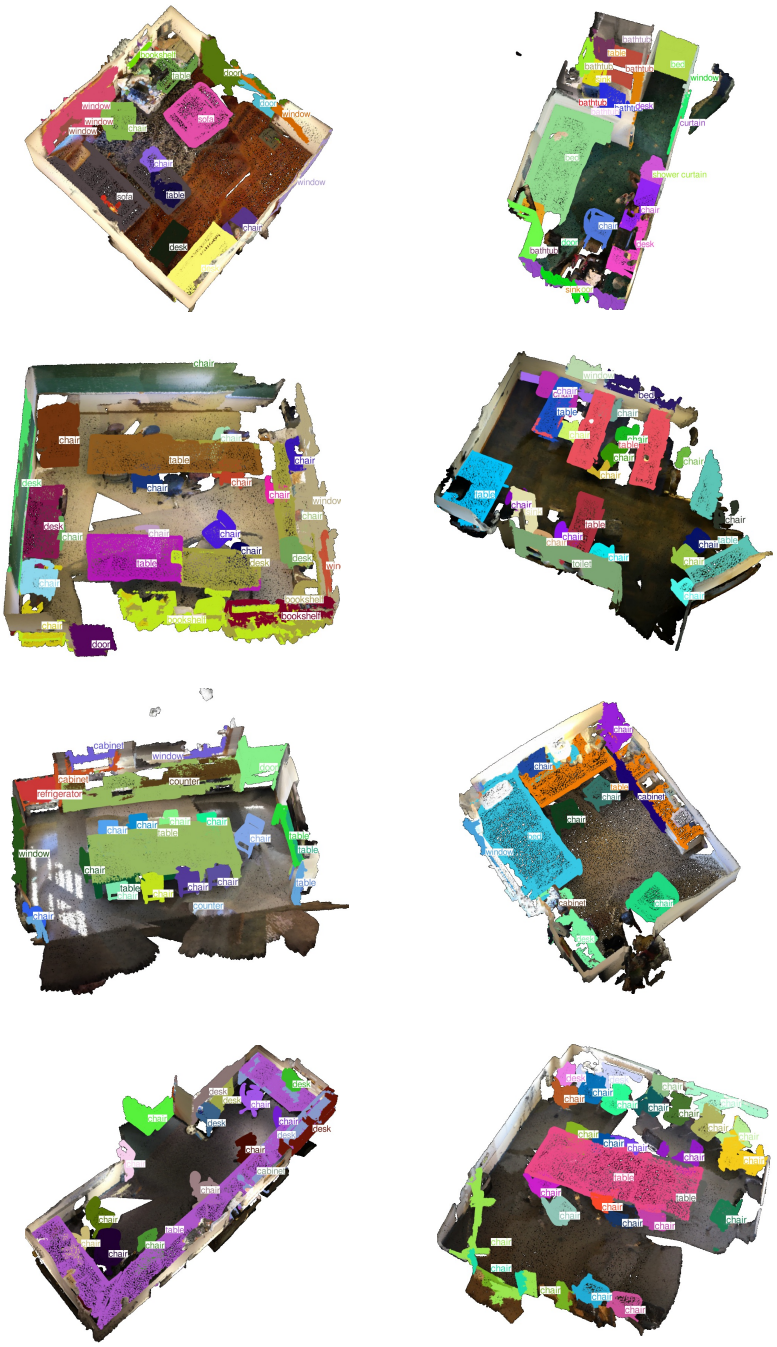


Figure 10: **Open-vocabulary Instance Segmentation Results of ScanNetv2 by OpenIns3D (ODISE).** Instance and class labels are presented in the same color.

Input Queries: *'building', 'vegetation', 'vehicle', 'truck', 'Aircraft', 'military vehicle', 'bike', 'motorcycle', 'light pole', 'street sign', 'clutter', 'fence'*

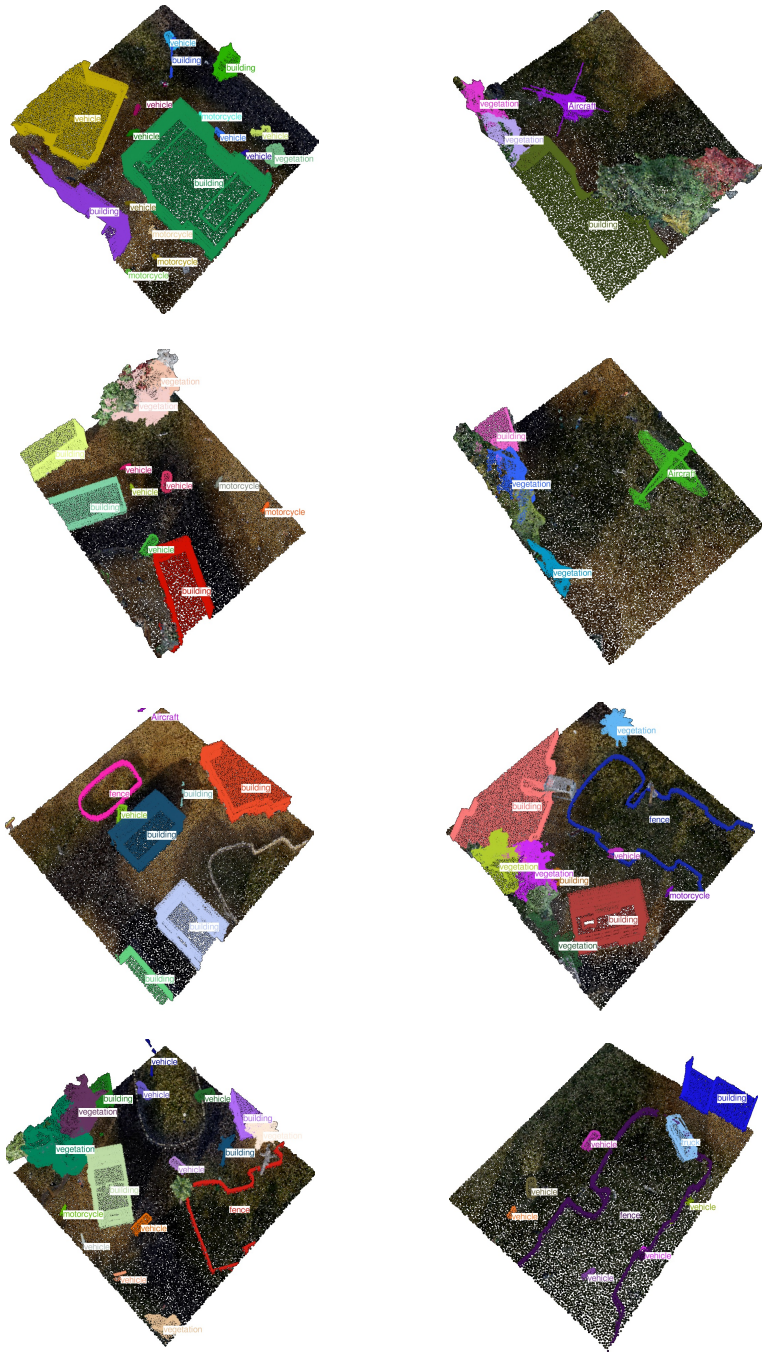


Figure 11: **Open-vocabulary Instance Segmentation Results of STPLS3D by OpenIns3D (ODISE).** Instance and class labels are presented in the same color.