Generalized Few-shot 3D Point Cloud Segmentation with Vision-Language Model
Supplementary Material

Zhaochong An[1,2], Guolei Sun[2*], Yun Liu[3*], Runjia Li[4], Junlin Han[4],
Ender Konukoglu[2], Serge Belongie[1]
[1] Department of Computer Science, University of Copenhagen
[2] Computer Vision Laboratory, ETH Zurich
[3] College of Computer Science, Nankai University
[4] Department of Engineering Science, University of Oxford

1. Additional Details on Novel-Base Mix

To effectively utilize support samples, the Novel-Base Mix approach is designed to integrate them into the base training inputs while preserving essential scene context. This ensures effective learning of challenging novel classes. We provide the pseudo-code for Novel-Base Mix in Algorithm 1. Below, we present a step-by-step explanation of the process.

Step 1. The process begins with cropping the region of novel objects from the novel point cloud. Given the *randomly sampled* novel point cloud and its corresponding binary mask, a cropping operation is applied to extract the relevant local region. This ensures that only the novel object region is considered for mixing, while extraneous unnecessary points are excluded.

Step 2. Next, to align the cropped novel sample with the base point cloud, we identify key spatial anchors in the *XY plane*. These anchors correspond to the top, bottom, left, and rightmost corner points of both the base point cloud and the cropped novel point cloud. These anchors serve as reference points for spatial alignment in subsequent steps.

Step 3–4. A random corner from {`top, bottom, left, right`} is selected for alignment. For instance:

- If the bottom corner is chosen, the lowest corner of the base point cloud is aligned with the highest corner of the novel point cloud.
- Conversely, if the left corner is selected, the leftmost corner of the base point cloud is aligned with the rightmost corner of the novel point cloud.

This pairing strategy introduces diversity in the placement of novel objects while ensuring the preservation of contextual integrity. Based on the selected corner pair, a translation vector is computed to spatially align the novel sample with the base point cloud.

Step 5–6. The computed translation vector is applied to the cropped novel point cloud, ensuring that it is positioned next to the base point cloud in the XY plane.

Additionally, Z-axis alignment is performed by adjusting the Z-coordinates of the translated novel point cloud. This step ensures that the novel sample is grounded at the same level as the base point cloud, preventing it from floating above or sinking below the base scene.

Step 7. Finally, the aligned novel sample is merged with the base point cloud to form the mixed training input. This effectively integrates the novel sample into the training scene while retaining its original context, which helps the model recognize complex and challenging novel classes.

We provide additional visualizations of the outputs from our Novel-Base Mix in Fig. 1.

2. Additional Details on the new Benchmarks

As discussed in Sec.5.1, we leverage two recent datasets, ScanNet200 [4] and ScanNet++ [8], to construct comprehensive evaluation benchmarks for GFS-PCS.

ScanNet200 [4] extends the labeling space of ScanNet [1] from 20 to 200 categories, introducing finer-grained subclasses of existing categories and numerous novel object types. These expansions enhance the dataset's granularity and diversity, making it a valuable resource for evaluating GFS-PCS methods. Meanwhile, ScanNet++ [8] offers annotations for 460 scenes encompassing over 1,000 unique object classes. This dataset captures a broad range of object categories, reflecting the complexity and variability of real-world environments. Together, these datasets form a rich and diverse foundation for constructing robust GFS-PCS evaluation benchmarks.

*Corresponding authors: Guolei Sun and Yun Liu

Algorithm 1: Pseudo-code of Novel-Base Mix in PyTorch style.

```
# Input:
# novel_cloud: point cloud of the novel class, shape (
     N, 3)
# novel_mask: binary mask for points belonging to the
     novel class, shape (N,)
# base_cloud: point cloud of base classes, shape (M,
     3)
# random_corner: function to randomly select a corner
     ('bottom', 'top', 'left', 'right')
# crop_fn: function to crop novel point clouds based
     on the mask
# corner_fn: function to compute corner points in the
     XY plane,
# returning a dictionary with keys: ['top', 'bottom',
     'left', 'right']

# Step 1: Crop the novel point cloud based on the mask
novel_local, novel_local_mask = crop_fn(novel_cloud,
     novel_mask)

# Step 2: Compute corner points for both point clouds
base_corners = corner_fn(base_cloud)
novel_corners = corner_fn(novel_local)

# Step 3: Randomly select a corner for alignment
selected_corner = random_corner(['bottom', 'top', '
     left', 'right'])

# Step 4: Calculate the translation vector based on
     selected corners
if selected_corner == "bottom":
   base_point = base_corners['bottom'] # Lowest point
        of base cloud in Y
   novel_point = novel_corners['top'] # Highest point
        of novel cloud in Y
elif selected_corner == "top":
   base_point = base_corners['top'] # Highest point of
         base cloud in Y
   novel_point = novel_corners['bottom'] # Lowest
        point of novel cloud in Y
elif selected_corner == "left":
   base_point = base_corners['left'] # Leftmost point
        of base cloud in X
   novel_point = novel_corners['right'] # Rightmost
        point of novel cloud in X
else: # "right"
   base_point = base_corners['right'] # Rightmost
        point of base cloud in X
   novel_point = novel_corners['left'] # Leftmost
        point of novel cloud in X

translation_vector = [base_point[0] - novel_point[0],
                      base_point[1] - novel_point[1],
                      0] # No z-translation yet

# Step 5: Translate the novel cloud in the XY plane
novel_local_translated = novel_local +
     translation_vector

# Step 6: Align the z-coordinates
z_adjustment = min(base_cloud[:, 2]) - min(
     novel_local_translated[:, 2])
novel_local_translated[:, 2] += z_adjustment

# Step 7: Combine base cloud and translated novel
     cloud
mixed_cloud = torch.cat([base_cloud,
     novel_local_translated], dim=0)
```

Benchmark Design. To create meaningful and representative GFS-PCS benchmarks, we carefully selected classes based on their occurrence counts in the respective datasets, ensuring sufficient representation across scenes. The selection process involved computing the occurrence count of each class across the dataset, ranking the classes by their occurrence counts, and assigning them to base and novel sets as follows:

- ScanNet200: Classes with occurrence counts exceeding 100 were retained, yielding 57 classes in total. The 12 most frequently occurring classes were designated as base classes, while the remaining 45 were assigned as novel classes.
- ScanNet++: Classes with occurrence counts exceeding 80 were retained, resulting in 30 classes in total. The top 12 most frequent classes formed the base class set, while the remaining 18 were assigned as novel classes.

These frequency thresholds were carefully chosen to strike a balance between class diversity and adequate representation, ensuring that both base and novel classes are well-suited for evaluating GFS-PCS performance.

Benchmark Classes. The following are the specific class lists for the two benchmarks:

- ScanNet200:
  - `Base Classes`: [‘refrigerator’, ‘desk’, ‘curtain’, ‘bookshelf’, ‘bed’, ‘table’, ‘window’, ‘cabinet’, ‘door’, ‘chair’, ‘floor’, ‘wall’]
  - `Novel Classes`: [‘trash can’, ‘ceiling’, ‘doorframe’, ‘object’, ‘shelf’, ‘sink’, ‘picture’, ‘backpack’, ‘couch’, ‘box’, ‘pillow’, ‘radiator’, ‘mirror’, ‘whiteboard’, ‘lamp’, ‘toilet’, ‘book’, ‘monitor’, ‘towel’, ‘tv’, ‘clothes’, ‘coffee table’, ‘office chair’, ‘nightstand’, ‘bag’, ‘dresser’, ‘toilet paper’, ‘recycling bin’, ‘kitchen cabinet’, ‘bathtub’, ‘telephone’, ‘plant’, ‘stool’, ‘keyboard’, ‘shoe’, ‘jacket’, ‘shower curtain’, ‘armchair’, ‘microwave’, ‘computer tower’, ‘bathroom vanity’, ‘kitchen counter’, ‘shower wall’, ‘paper towel dispenser’, ‘file cabinet’]
- ScanNet++:
  - `Base Classes`: [‘wall’, ‘floor’, ‘door’, ‘ceiling’, ‘table’, ‘window’, ‘box’, ‘ceiling lamp’, ‘light switch’, ‘cabinet’, ‘chair’, ‘heater’]
  - `Novel Classes`: [‘monitor’, ‘whiteboard’, ‘office chair’, ‘bottle’, ‘doorframe’, ‘keyboard’, ‘window frame’, ‘mouse’, ‘paper’, ‘blinds’, ‘trash can’, ‘telephone’, ‘book’, ‘shelf’, ‘sink’, ‘windowsill’, ‘bag’, ‘smoke detector’]

Overall, our benchmarks provide a more robust and comprehensive testbed for evaluating GFS-PCS methods. By better reflecting real-world challenges, our benchmarks enable researchers to rigorously assess models’ performance and generalization to novel categories under realistic scenarios.

3. Additional Implementation Details

Our framework employs a straightforward segmentor consisting of a backbone and a linear classification head, designed for both efficiency and simplicity to facilitate reproducibility. The training process consists of two stages:

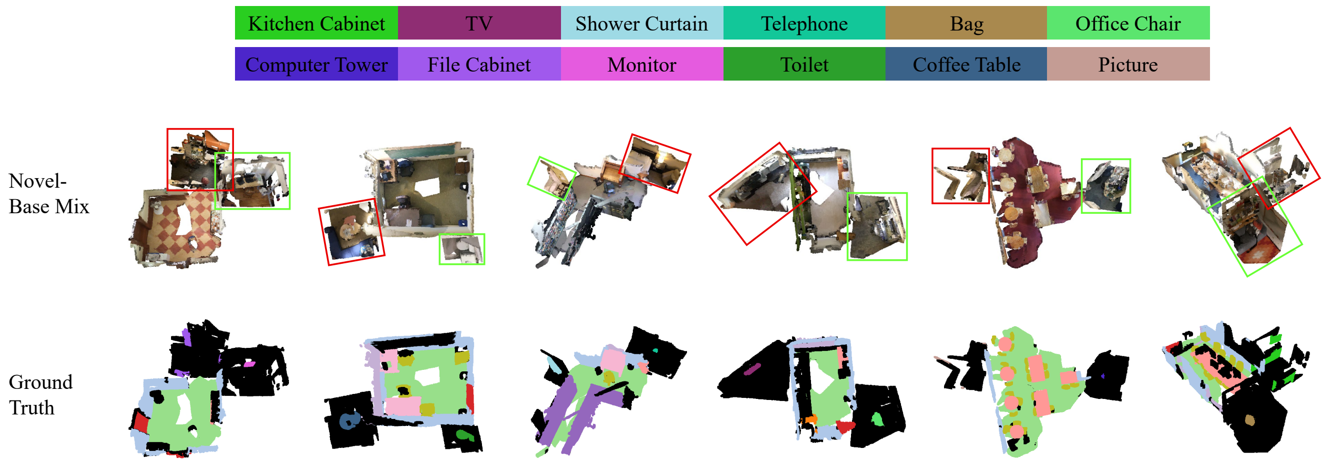

Figure 1. Visualization of the outputs from the proposed Novel-Base Mix. The red and green boxes represent the two novel samples mixed into the scene. The novel class colors are shown at the top.

pretraining on the base classes of each dataset, followed by fine-tuning with adding a separate linear classification head for novel classes. For prompting the 3D VLMs, we adopt the default prompt used in RegionPLC [7] and OpenScene [3]: "a CLASS_NAME in a scene". We evaluate two widely used backbones in our experiments: Point Transformer V3 (PTv3) [5] and SparseConvNet (SCN) [2]. All experiments were conducted using 4 NVIDIA RTX 4090 GPUs.

For pretraining, we adhere to the default configurations provided in [5]. When using PTv3 as the backbone, the model is trained for 800 epochs with the AdamW optimizer. The learning rate is set to 0.006, with a reduced learning rate of 0.0006 for the backbone blocks, and a weight decay of 0.05. The OneCycleLR scheduler is employed to adjust the learning rate during training. When using SCN as the backbone, the model is trained for 600 epochs on ScanNet200 and 800 epochs on ScanNet++ and ScanNet. The SGD optimizer is employed, with a learning rate of 0.05 and a weight decay of 0.0001. Similar to PTv3, the learning rate is scheduled using the OneCycleLR strategy.

For fine-tuning, the network is trained for 20 epochs end-to-end with the Adam optimizer. A constant learning rate is used, with a value of 0.001 for ScanNet200 and ScanNet, and 0.007 for ScanNet++. The backbone learning rate is reduced by a factor of 0.1 to stabilize training.

During training, the preprocessing follows the steps outlined in [5]. The raw input points are voxelized using a grid size of 0.02m, and a random cropping operation is applied to ensure that the number of points in each training input remains within a maximum limit, such as 102,400 points.

In the evaluation phase, the input point clouds only undergo voxelization without any further cropping or sampling operations. This enables testing on full scenes, as opposed to small blocks used in previous GFS-PCS evaluations [6, 9]. By evaluating on entire scenes, it better simulates real-world scenarios and provides a more realistic and comprehensive assessment of models' performance.

4. Additional Visualizations

In this section, we present additional qualitative results to further illustrate the efficacy of our approach in addressing GFS-PCS tasks. These results highlight the superiority of our model in novel class generalization and segmentation quality, providing deeper insights into the design and impact of our proposed modules.

Comparison with State-of-the-Art Methods. Figure 2 showcases additional segmentation results comparing our proposed framework, GFS-VL, against the previously established state-of-the-art method, GW [6], on the ScanNet200 [4] benchmark. For clarity, class colors used in the visualizations are displayed on the right side of the figure and are restricted to those present in the ground truth.

These visualizations clearly demonstrate the superior performance of GFS-VL, which effectively integrates dense semantic knowledge embedded in 3D VLMs with precise guidance from few-shot samples. This synergy enables GFS-VL to achieve robust novel class generalization in the challenging benchmarks. The qualitative results highlight improved boundary delineation, more accurate segmentation of novel objects, and a better overall alignment with ground truth.

Despite achieving better performance, Figure 2 also exposes some limitations. Specifically, our model exhibits suboptimal performance on small objects (*e.g.*, *Trash Can* in the third row), thin objects (*e.g.*, *Curtain* in the third row), and objects within complex backgrounds (*e.g.*, *Bath-*

room Vanity in the first row). Addressing these challenges presents promising directions for future work.

Improvements in Pseudo-label Quality. In Figure 3, we provide additional visualizations of the refinement process for raw pseudo-labels, illustrating the role of our Pseudo-label Selection (PS) and Adaptive Infilling (AI) modules.

- PS filters noisy predictions from the 3D VLM by anchoring them to the accurate few-shot samples, ensuring high reliability.
- AI discovers novel objects that were initially missed in the raw pseudo-labels, as indicated in the red circles in Figure 3, and completes partially segmented regions, as shown in the green circles.

By integrating few-shot support samples with the current pseudo-label context, the AI module creates adaptive prototypes that facilitate both the discovery of missed novel objects and the completion of partial pseudo-labels, thereby enhancing the quality of novel region labels. Together, PS and AI play distinct yet complementary roles in pseudo-label refinement. By combining dense knowledge from 3D VLMs with the precision of few-shot samples, these modules significantly improve pseudo-label quality, achieving better alignment with the full-class ground truth.

For visualization, the class colors in Figure 3 are displayed at the top and correspond to labels present in the full-class annotations.

References

[1] 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. 1

[2] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232, 2018. 3

[3] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. OpenScene: 3D scene understanding with open vocabularies. In *CVPR*, pages 815–824, 2023. 3

[4] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3D semantic segmentation in the wild. In *ECCV*, pages 125–141, 2022. 1, 3

[5] 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. 3

[6] Yating Xu, Conghui Hu, Na Zhao, and Gim Hee Lee. Generalized few-shot point cloud segmentation via geometric words. In *ICCV*, pages 21506–21515, 2023. 3, 5

[7] Jihan Yang, Runyu Ding, Weipeng Deng, Zhe Wang, and Xiaojuan Qi. RegionPLC: Regional point-language contrastive learning for open-world 3D scene understanding. In *CVPR*, pages 19823–19832, 2024. 3

[8] 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. 1

[9] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Few-shot 3D point cloud semantic segmentation. In *CVPR*, pages 8873–8882, 2021. 3

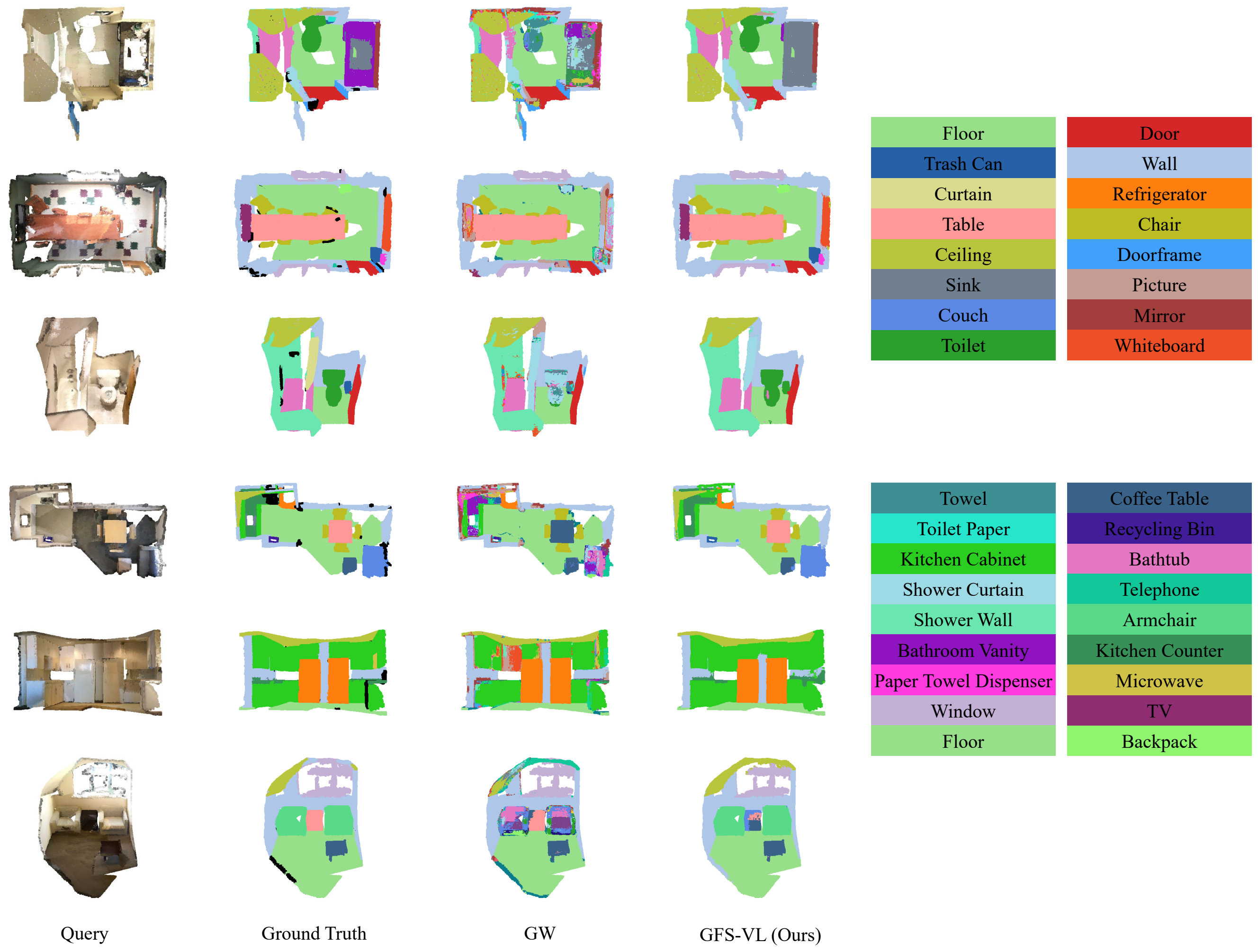

Figure 2. **Qualitative comparison between GW [6] and our GFS-VL on ScanNet200.** The visualizations demonstrate the superior segmentation performance and novel class generalization capabilities of GFS-VL. For clarity, class colors are displayed on the right and are restricted to those present in the ground truth annotations.

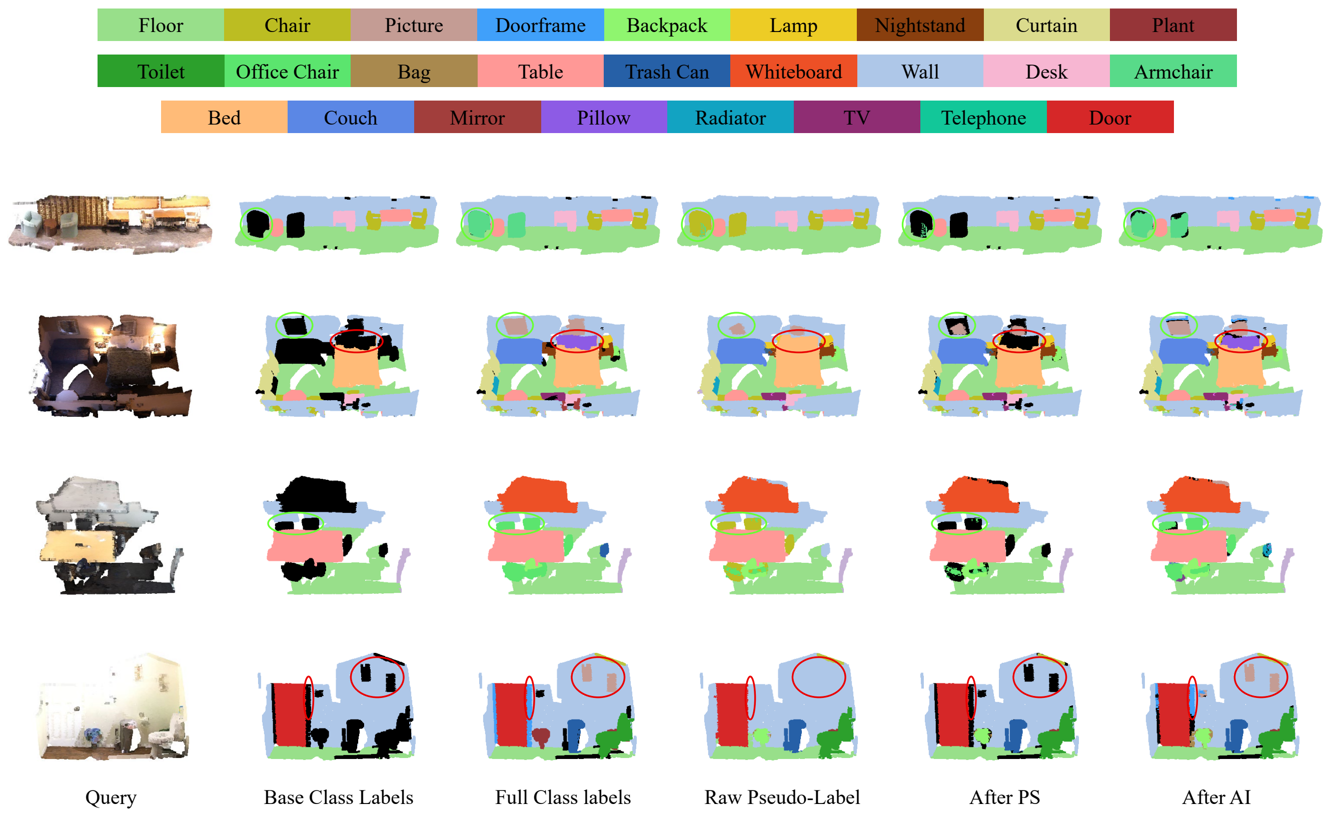

Figure 3. **Visualization of pseudo-label refinement using Pseudo-label Selection (PS) and Adaptive Infilling (AI).** Red circles indicate novel objects discovered by AI that were missed in the raw pseudo-labels, while green circles indicate regions where AI completes previously partially segmented areas. For clarity, class colors are displayed at the top and correspond to labels present in the full class annotations.