Learning Point Cloud Completion without Complete Point Clouds: A Pose-Aware Approach
–Supplementary materials–

Jihun Kim, Hyeokjun Kwon, Yunseo Yang, and Kuk-Jin Yoon
Korea Advanced Institute of Science and Technology
{jihun1998,0327june,acorn,kjyoon}@kaist.ac.kr

1. Implementation Details

1.1. Architecture Details

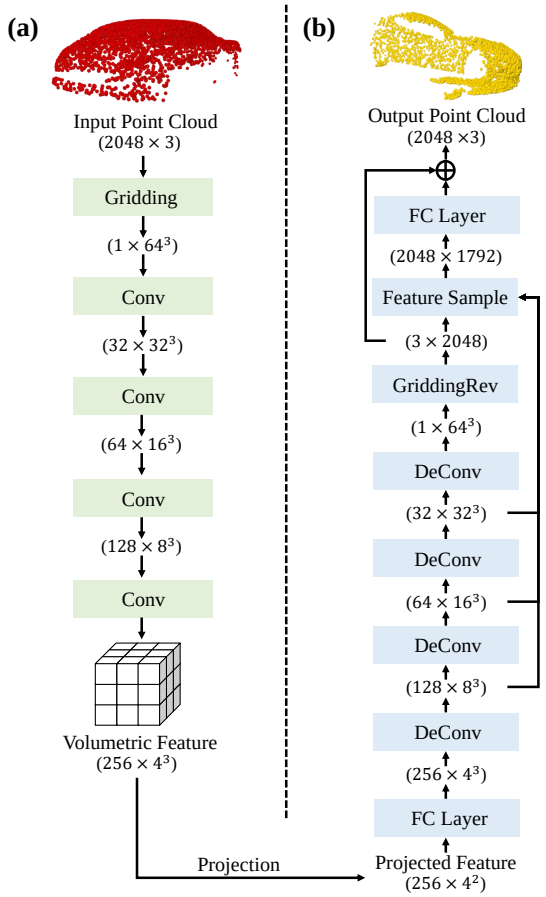

Figure 1: Detailed architectures of encoder and decoder. (a) In the encoder, volumetric feature is obtained by passing "Gridding", 4 "Conv" blocks. Then, the projected feature is obtained from the projection method. (b) In the decoder, the final output incomplete point cloud is generated by decoding the projected feature with "FC Layers", 4 "DeConv" blocks, "GriddingRev", and "Feature Sample" process.

The details of the encoder architecture are shown in Fig. 1 (a). We follow some modules from GRNet [4] to apply a gridding structure and volumetric feature. We use the gridding module proposed in GRNet to create a gridded voxel representation of the input point cloud. The dimension of the gridded voxel representation is 64^3. The "Conv" block in Fig. 1 (a) includes a 3D convolutional layer with a kernel size of 4 and padding of 2, a batch normalization, a leaky ReLU activation, and a max pooling layer with a kernel size of 2. After the gridded voxel is entered, the volumetric feature with dimension of 256×4^3 is extracted from 4 Conv blocks. Finally, a projected feature with dimension of 256×4^2 is obtained from the volumetric feature using the proposed projection method conditioned by a given pose (ϕ, θ).

Next, the details of the decoder architecture are shown in Fig. 1 (b). The projected feature is entered into "FC Layer" first. The "DeConv" block in Fig. 1 (b) includes a 3D transposed convolutional layer with a kernel size of 4 and stride of 2, a batch normalization, and a ReLU activation. After 4 DeConv blocks, intermediate gridding is generated with 64^3 size. We also use the "GriddingRev" module and "Feature Sample" module proposed in GRNet. Each feature obtained from 4 DeConv blocks is used in the Feature Sample module. After passing through the decoder, the output point cloud with dimension of 2048×3 is generated from the projected feature.

1.2. Details on Projection Method

In the main paper, we omit some details about the proposed projection method. One of these details is the process of determining the base region according to the pose, as shown in Fig. 2 (a). We first select a center point by computing ϕ_{center} and θ_{center} as follow:

$$\phi_{center} = round(\phi/30°) \times \phi \quad (1)$$

$$\theta_{center} = round(\theta/30°) \times \theta. \quad (2)$$

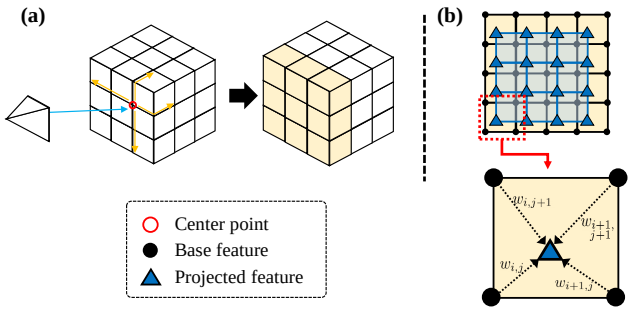

Figure 2: Illustration of the detailed process involved in the projection method. (a) The center point is selected, and the 5×5 base region is determined around it. (b) Bilinear interpolation is applied in the projected feature by calculating the weights for each neighboring base feature according to the pose.

Here, the $round$ function rounds the value in parentheses to the nearest integer. Since the interval between each grid is regarded as 30° in the volumetric feature, the center point can be selected based on ϕ_{center} and θ_{center}. Finally, using the center point as the reference, the base region can be determined as shown in Fig. 2 (a).

The next part is the process of obtaining the projected feature from the base feature, which is shown in Fig. 2 (b). To apply bilinear interpolation, we calculate $\phi_{remainder}$ and $\theta_{remainder}$ as follow:

$$\phi_{remainder} = \phi \,\%\, 30^\circ \tag{3}$$
$$\theta_{remainder} = \theta \,\%\, 30^\circ. \tag{4}$$

Here, $\%$ function returns the remainder of a division operation. Using these values, weights shown in Fig. 2 (b) can be calculated with the following equations:

$$r_\phi = \phi_{remainder}/30^\circ,\ r_\theta = \theta_{remainder}/30^\circ \tag{5}$$
$$w_{i,j} = (1 - r_\phi)(1 - r_\theta) \tag{6}$$
$$w_{i+1,j} = r_\phi(1 - r_\theta) \tag{7}$$
$$w_{i,j+1} = (1 - r_\phi)r_\theta \tag{8}$$
$$w_{i+1,j+1} = r_\phi r_\theta. \tag{9}$$

Now, projected features can be calculated from the weights and the neighboring base features.

1.3. Details on Dataset Generation

The dataset generation process involves utilizing the technique described in [2] to eliminate unobserved points from a given point cloud with a specific pose. Since we have complete point clouds from ShapeNet [1], we generate incomplete point clouds that would be used for input. This ensures that the generated point clouds are realistic and representative of real-world scenarios. To create incomplete point clouds for training and testing, poses are randomly assigned to each complete point cloud. The poses are expressed in spherical coordinates, with a radius of 1.5. The values of θ and ϕ are sampled from uniform distributions in the ranges of $[0, 180^\circ]$ and $[0, 360^\circ]$, respectively.

Figure 4 depicts the outcomes of the dataset generation process, including an example of an incomplete point cloud generated for each category. The second and fourth columns show incomplete point clouds that we generated with a random pose. The first two columns exhibit the object's point cloud from a designated viewpoint with $\theta = 45^\circ$ and $\phi = 45^\circ$. The self-occluded parts of the generated incomplete point cloud's missing portions can be seen in the second column when compared to the complete point cloud in the first column. The third and fourth columns exhibit the outcomes viewed from the pose employed in creating the incomplete point cloud. If the incomplete point cloud is well-generated for the intended pose, it should appear identical to the complete point cloud from that viewpoint. As expected, the shapes of point clouds in the third and fourth columns are similar, indicating that only unobservable points were removed based on the corresponding poses. Consequently, the incomplete point clouds are effectively generated and representative of real-world scenarios.

2. More Experiments

2.1. More Qualitative Comparisons

Figure 5 presents additional completion results, which include examples from previous studies [3, 5], to further evaluate the effectiveness of our proposed method. For each category, two different examples are shown consecutively. Our results demonstrate that the proposed method achieves comparable and sometimes even better completion performance across all categories when compared to previous studies with unpaired settings. This highlights the effectiveness of our proposed method for point cloud completion.

2.2. Results with Different Voxel Resolution

We verify the impact of voxel resolution on the performance. Table 1 shows the result of performance, FLOPs, and memory usage for resolutions of $3 \times 3 \times 3$, $4 \times 4 \times 4$, and $5 \times 5 \times 5$. As the voxel resolution increases, CD decreases, and F1-score increases. However, we can see that using a $5 \times 5 \times 5$ voxel resolution leads to a significant increase in both FLOPs and memory requirements. Taking this trade-off into account, we set voxel resolution as $4 \times 4 \times 4$ in our paper.

2.3. Effect of Internal Voxel Features

While we acknowledge that our rendering process utilizes the surface voxels only, we would like to clarify that the feature extraction is conducted using the internal voxels

Table 1: Impact of voxel resolution on performance. CD is Chamfer Distance scaled with $\times 10^4$. Best results are indicated as **bold**

Resolution	CD ↓	F1-score ↑	FLOPs (G) ↓	Memory (G) ↓
$3 \times 3 \times 3$	10.36	86.83	10.31	28.24
$4 \times 4 \times 4$	9.63	88.01	12.94	46.68
$5 \times 5 \times 5$	9.25	88.28	17.16	81.73

together. As shown in Fig. 1, the encoder processes the input with 3D convolutions, and the internal voxels are used for feature extraction. To prove this, we train our method with surface voxels only, omitting the internal voxels. The exclusion of internal voxels results in an increase of 0.6 CD and a decrease of 0.93 F1-score, demonstrating decreased performance.

2.4. Results with Other Types of Occlusion

We mainly focus on the incomplete point clouds caused by self-occlusion since it is an inherent limitation when acquiring point cloud data using scanners like LiDAR. However, we acknowledge that other types of occlusions, such as when other objects occlude the target object, can also occur. To address this issue, we conduct additional experiments with simulated occlusion. Specifically, we remove certain portions of the point cloud as if they were occluded by other objects. The substantial completion results depicted in Fig. 3 demonstrate the robustness of our method to other types of occlusion. We believe that this generalization capability is mainly due to the re-training phase, which enables the model to learn the concept of "completion" on multiple categories.

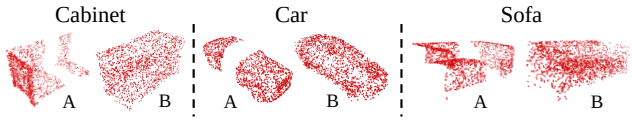

Figure 3: **A**: Input with simulated occlusion, **B**: Our result.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[2] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, pages 24–es. 2007. 2

[3] Xin Wen, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13080–13089, 2021. 2, 5

[4] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020. 1

[5] Junzhe Zhang, Xinyi Chen, Zhongang Cai, Liang Pan, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. Unsupervised 3d shape completion through gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1768–1777, 2021. 2, 5

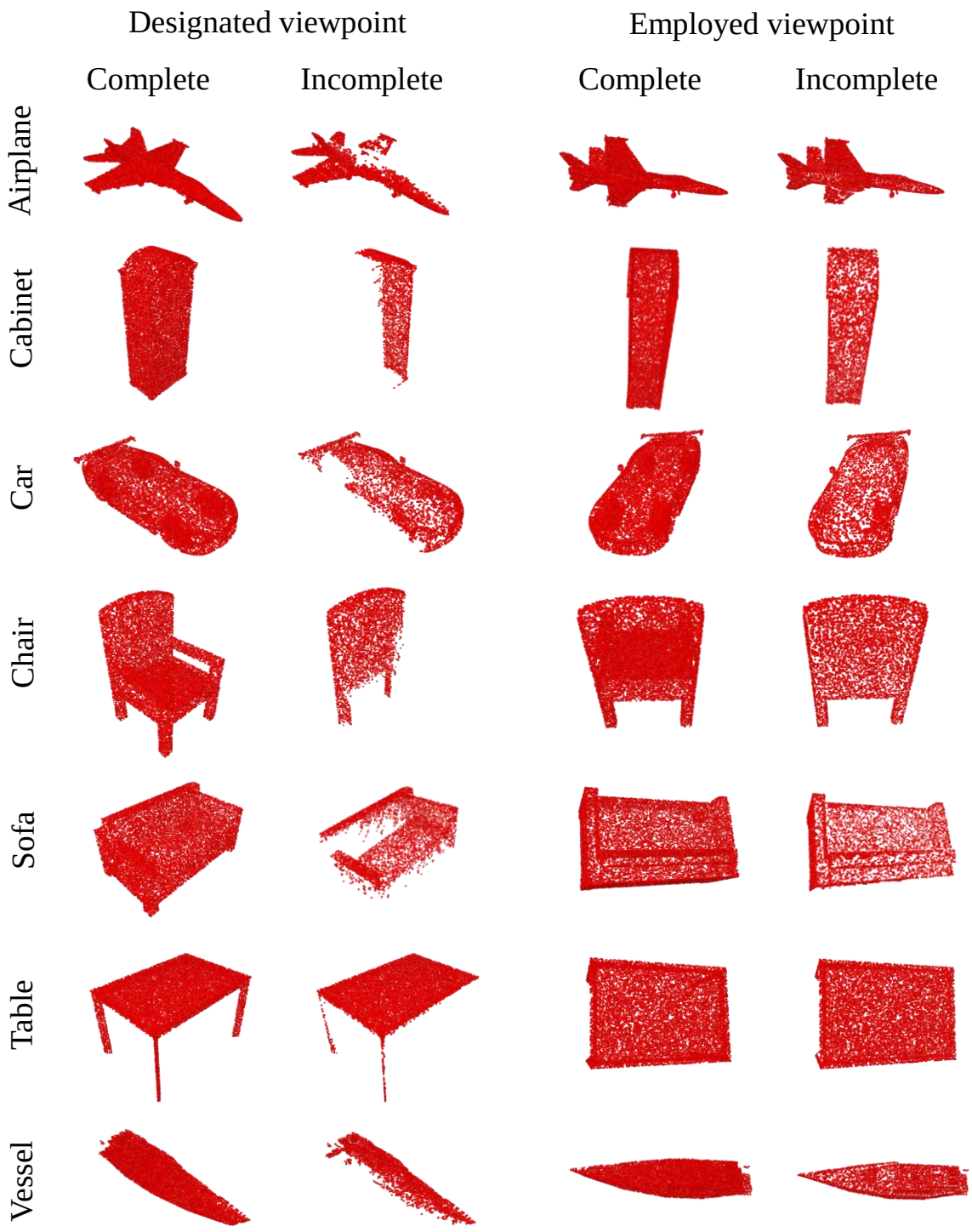

Figure 4: Visualization of generated incomplete point clouds. Point clouds in the first and second columns are all viewed from the same fixed pose. Point clouds in the third and fourth columns are viewed from the pose, which is used for generating incomplete point clouds. Incomplete point clouds in the second and fourth columns are generated by eliminating unobserved points.

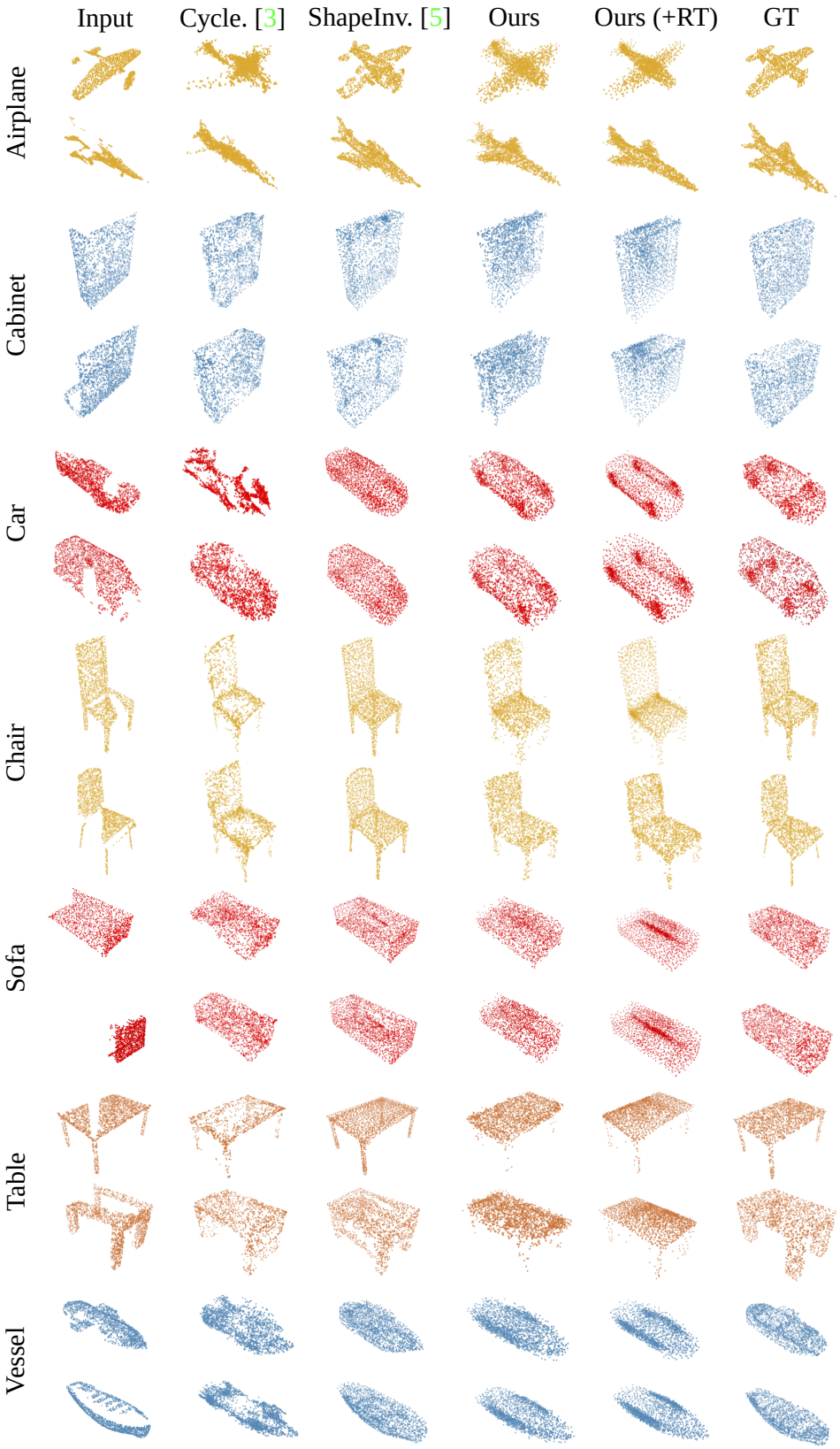

Figure 5: Visualization of completion results for all categories. From left to right: incomplete inputs, results of Cycle4completion [3], ShapeInversion [5], ours without RT, ours with RT, and GT.