Supplementary: Learning Semantic Part-Based Graph Structure for 3D Point Cloud Domain Generalization

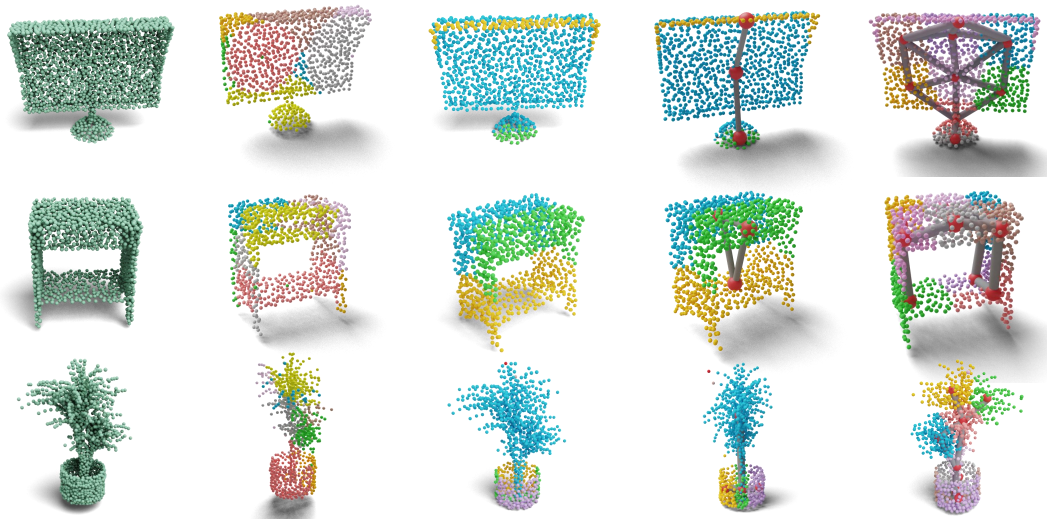

Figure 1. Input Point Clouds (1st Column), Result of directly applying FPS to inputs (we divide them into 8 parts as PDG to show qualitative comparison) (2nd Column), Outputs of our proposed Unsupervised Part Decomposition (UPD) module (3rd Column), Output of Graph Structure Induction (GSI) module without fine-grained partitions (FPS parts) (4th Column) and Output of rich Graph Structure Induction (GSI) module with fine-grained partitions (FPS parts) (5th Column). Different colours in the 2nd, 3rd, 4th, and 5th columns signify different parts of the same point cloud. Grey lines in the 4th and 5th columns signify connections between parts created by the GSI module; vertices are shown in red.

A. Qualitative Analysis

This section presents additional qualitative results of the outputs of our UPD and GSI modules, illustrating their utility. Figure 1 displays the outputs of both modules.

B. Automatic vs Fixed Number of Clusters:

Settings	M-S	M-S*	Avg.
UPD: Fixed 4 Clusters	84.49	55.13	69.81
UPD: Fixed 6 Clusters	84.21	54.51	69.36
UPD: Fixed 8 Clusters	85.04	55.35	70.20
UPD: Fixed 10 Clusters	86.19	58.74	72.47
UPD: Fixed 12 Clusters	86.39	58.91	72.65
UPD: Automatic Clusters	**86.45**	**59.49**	**72.97**

Table 1. Study the impact of automatic clustering vs Fixed number of clusters (M as source)

We conduct experiments to understand the efficiency of our optimization framework for cluster number in UPD by comparing it with *viz.* fixed clustering applied to the UPD module. Table 1 shows the effectiveness of automatic clustering based on object geometry in contrast to fixed numbers of clusters. In our experiments, the fixed clusters ranged from 4 to 12, with 2 clusters increased per experiment. After completing part segmentation by the UPD module, we divide these parts into smaller sub-parts using FPS, as discussed in Section 3.2 of the main paper, hence effectively increasing the number of sub-parts beyond the number of clusters. Here we choose a subset of our original experiments where we understand the effect of automatic clustering in both the *Simulated to Simulated* scenario ($M \rightarrow S$) and the *Simulated to Real* scenario ($M \rightarrow S^*$).

Choosing the optimal number of parts: Table 1 shows that increasing the number of clusters helps improve the performance closer to our achieved State-of-the-art results. However, since we set 2 to 6 clusters in the UPD module, it takes a significantly higher number of rigid/fixed clusters

to even approach the achieved results, leading to more computing and memory consumption.

We also observe that increasing the clusters rigidly positively affects the results up to a specific limit. The main reason these experiments cannot achieve or surpass the state-of-the-art is that some objects require a lesser number of clusters to segment them into different parts meaningfully, and fixing a significantly higher number of clusters leads to the problem of over-segmentation (Note that, after the parts are segmented, they undergo FPS to be further divided into smaller sub-parts).

Hence, the hierarchical geometric structure is lost, and the GSI module can no longer understand the graph structure created from these over-segmented parts. Therefore, through these experiments, we know the importance of optimally choosing the correct number of clusters for different objects, even within the same class. We also observe that some objects, even within the same class, can be relatively simple or complex based on their geometric appearance. UPD module correctly handles individual objects within each class and optimally divides them into different numbers of clusters, leading to better part segmentation and, consequently, better graph creation.

C. Rerun on GraspNet-PC 10 Dataset:

We conduct three different runs of our proposed method on the GraspNet-PC 10 dataset to understand the statistical significance of the performance. In Table 2, we provide the result of the reruns. All the reruns use the same experimental configurations mentioned in the main paper's implementation details section. We can observe from the three runs that the results are statistically significant and consistent across different runs.

Experiment	Syn→Kin	Syn→RS	Kin→RS	RS→Kin
Run - 1	96.92	87.46	87.04	71.14
Run - 2	96.09	86.05	87.71	71.23
Run - 3	96.28	87.46	87.95	70.97
Average	96.43	86.99	87.57	71.12

Table 2. Rerun of proposed method on GraspNetPC-1o dataset. Classification accuracy (%) on the GraspNet-10. Sys.: Synthetic domain, Kin.: Kinect domain, RS.: RealSense domain.

D. Discussion on performance in S*→M and RS→Kin:

ScanNet (S*) and RealSense (RS) are real-world datasets captured using LiDAR sensors. S* is especially popular in DA and DG communities for their difficulties (viz. incomplete scans, hard-to-recognize objects). Hence, the parts and graphs created from S* and RS are very different. Still, the results obtained by our proposed method show its capabilities in handling complex source data. For these reasons, these experimental settings fail to surpass the current State-of-the-Art but achieve comparable performance (Refer to Table 2 in main paper).