Patch-based Progressive 3D Point Set Upsampling (Supplemental Material)

This supplementary material provides further technical details and experimental results on an extended dataset (scanned data and ModelNet10).

Implementation Details. By default, we use $16\times$-upsampling models with four $2\times$ upsampling units. In each upsampling unit, we implement 4 dense blocks containing 2 dense layers. The output size of the dense layers G is set to 12. The compression layer reduces the input prior of each dense block to a fixed number of features C', and we set $C' = 24$. The feature-based KNN inside the dense blocks uses $K = 32$, while the KNN used during feature interpolation is with $K = 5$.

The input patches for each upsampling unit have a fixed size N. The choice of N depends on the size of input shape. For the MNIST contour dataset, $N = 50$; for the 3D dataset, $N = \min(312, |P_0|)$, where $|P_0|$ is the size of the initial input.

For all experiments, we use a batch size of 28 and a fixed learning rate of 0.001. Adam optimizer with $\beta = 0.9$ is used for optimization.

Training 450 epochs using the Sketchfab dataset takes 7 hours on a single GeForce GTX 1080 Ti GPU. Inference time depends on the number of points in the initial input, the patch size, as well as the number of patches to be extracted in each level. Typically, performing $16\times$ upsampling from 625 and 5000 input points, as shown in Figure 10 and 11 in the main paper, takes about 5 and 18 seconds, respectively.

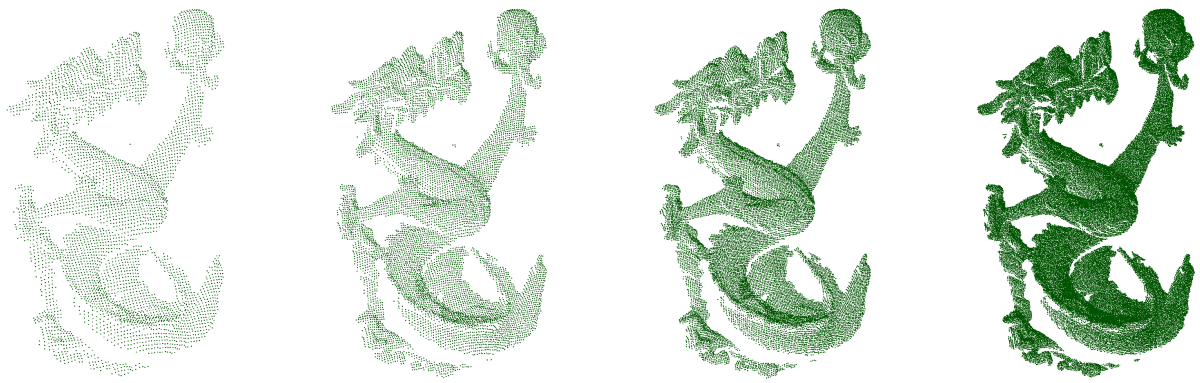

Figure 1: Examples of generated virtual scans. Such scans are used as training and testing data.

Ablation study for the feature extraction unit. We conduct experiments to validate the design choice of our feature extraction unit. The baseline of this study is the PointNet++ [1] inspired feature extractor used in PU-Net [2] with reduced width, such that the model size is comparable with ours. We incrementally add one component of our feature extractor and evaluate the chamfer distance and hausdorff distance. As shown in Table 1, all components consistently improvement the upsampling quality in both metrics. When combined together, our feature extractor significantly outperforms the baseline model with fewer parameters.

Upsampling of scanned data. We extend our model to handle scanned data by synthesizing a large amount of training data similar to real scans. We implement a virtual scanner to scan the training set of the Sketchfab dataset from a random

	feature extraction unit	CD $\cdot 10^{-3}$	HD $\cdot 10^{-3}$	# Params
1)	narrow PU-Net	0.93	11.47	513K
2)	1) + feature-based kNN	0.86	9.83	554K
3)	2) + w/o subsample	0.62	7.89	387K
4)	3) + dense links	0.54	6.92	304K

Table 1: Ablation study for the feature extraction unit. The baseline is the PointNet++ based feature extractor used in PU-Net; we reduced the width of this network for comparable network capacity.

viewpoint using different camera resolutions. By gradually increasing the camera resolution, we obtain a list of virtual scans with increasing density, as shown in Figure 1.

We test the trained model on both virtual and real scans. The results are shown in Figure 2 and 3. The virtual scans are generated from the testing set of the Sketchfab dataset, while the real scans are acquired via the Intel RealSense SR300 scanner. As an alternative to the WLOP consolidation, we use the scanner's built-in filter to obtain a relatively clean input, which typically smears the fine-grained details and reduces the frame rate. As shown in Figure 2, our model is able to reveal fine-grained geometry from very sparse inputs. In addition, it can fill holes and preserve sharp features, as shown in Figure 3.

Upsampling on ModelNet10. In addition to the quantitative evaluation on ModelNet10 shown in the main paper, we provide some examples for qualitative comparision.

As shown in Figure 4, we randomly pick one model from each category as input. The output points are color-coded based on the nearest neighbor distance to the ground truth. Similar to what is demonstrated in Table 1 of the main paper, EC-Net has difficulty handling sparse inputs. Overall, our model significantly outperforms PU-Net and EC-Net in this experiment.

References

[1] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *In Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017. 1

[2] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. Pu-net: Point cloud upsampling network. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 2790–2799, 2018. 1

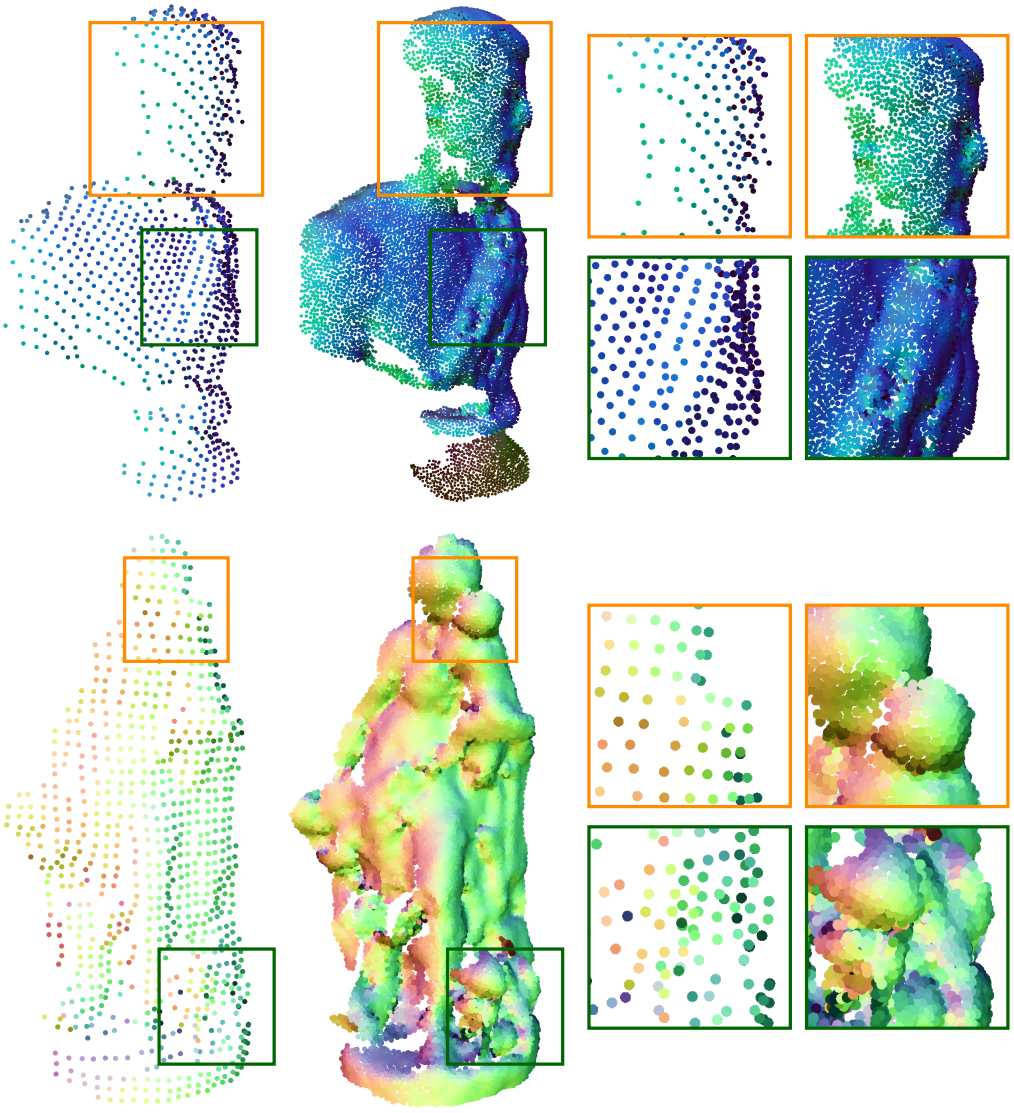

Figure 2: 16× upsampling results on virtual scans.

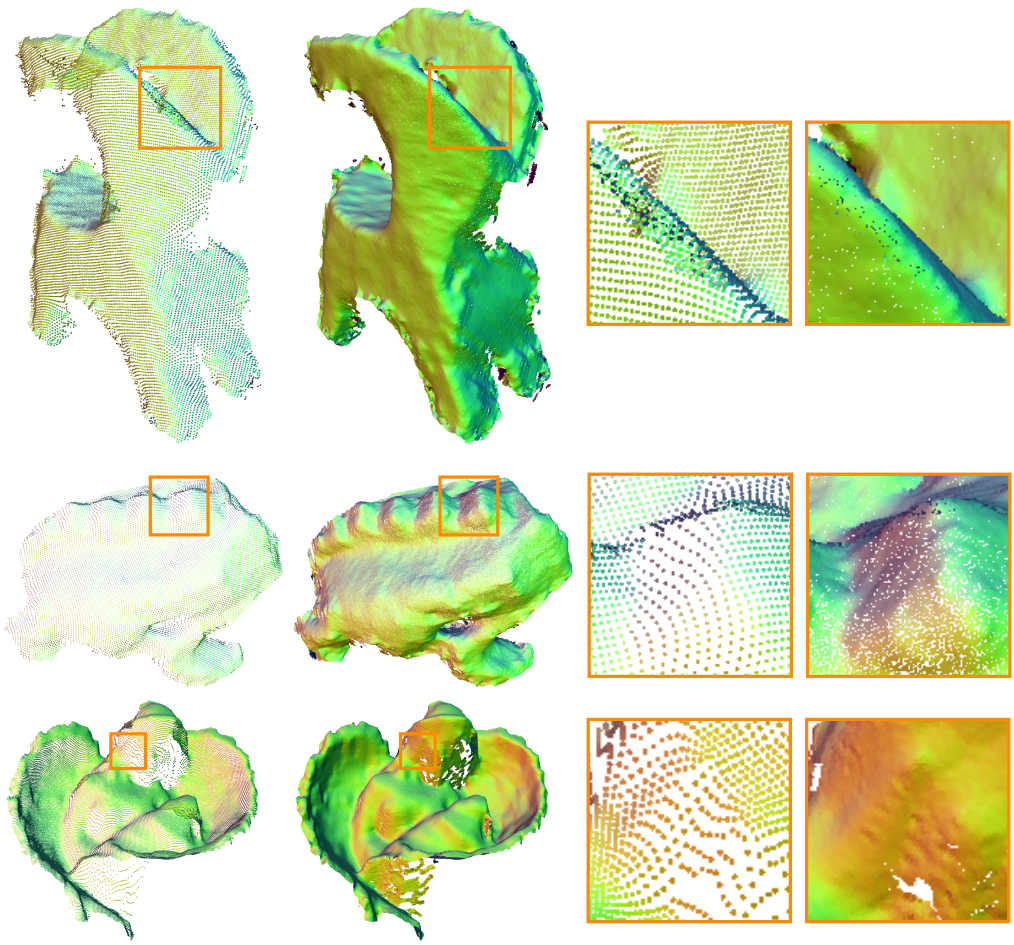

Figure 3: 16× upsampling results on real scans.

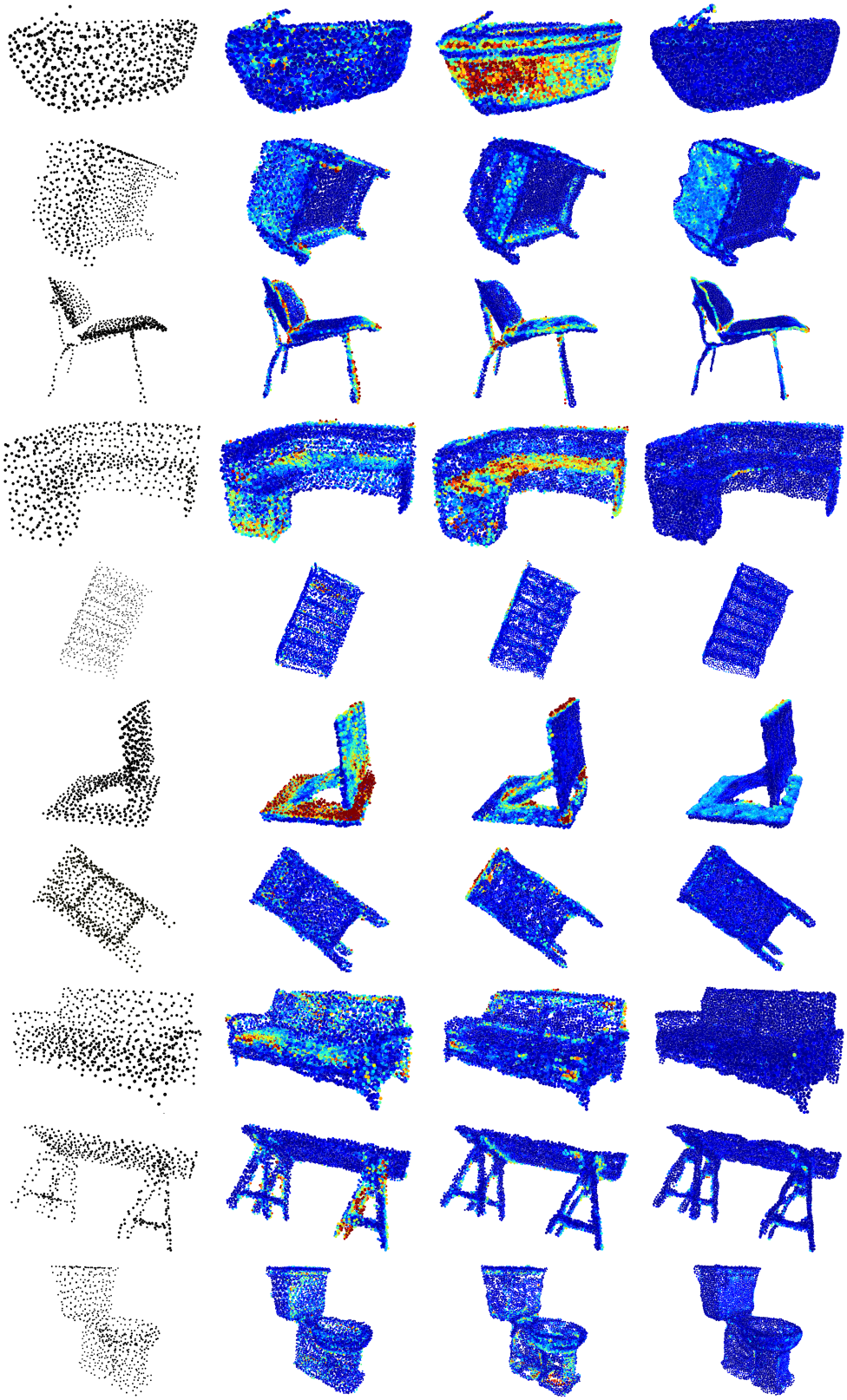

Figure 4: Upsampled examples of ModelNet10. The color encodes the distance to the closest neighbor in ground truth.