Data-driven Upsampling of Point Clouds

Wentai Zhang, Haoliang Jiang[1], Zhangsihao Yang[1], Soji Yamakawa, Kenji Shimada, Levent Burak Kara*

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

Abstract

High quality upsampling of sparse 3D point clouds is critically useful for a wide range of geometric operations such as reconstruction, rendering, meshing, and analysis. In this paper, we propose a data-driven algorithm that enables an upsampling of 3D point clouds without the need for hard-coded rules. Our approach uses a deep network with Chamfer distance as the loss function, capable of learning the latent features in point clouds belonging to different object categories. We evaluate our algorithm across different amplification factors, with upsampling learned and performed on objects belonging to the same category as well as different categories. We also explore the desirable characteristics of input point clouds as a function of the distribution of the point samples. Finally, we demonstrate the performance of our algorithm in single-category training versus multi-category training scenarios. The final proposed model is compared against a baseline, optimization-based upsampling method. Results indicate that our algorithm is capable of generating more uniform and accurate upsamplings.

Keywords:
point cloud, upsampling, deep learing, neural network

1. Introduction

With the emergence of 3D depth sensing technology, point cloud capture have become increasingly common in many applications involving shape digitization and reconstruction. While point cloud quality and point density have a critical impact on the subsequent digital design and processing steps, the large variety of sensing technologies coupled with varying characteristics of the object surfaces and the environment makes high-quality point cloud capture a difficult task. As such, to aid in reconstruction, digitally upsampling an input point cloud to produce a denser representation that remains true to the underlying object is a very desirable capability. However, it remains difficult to do so due to the need to *add* information that does not exists in the input.

Given a point cloud, two common approaches to reconstruction involve direct triangulation, and patch or field regression. Amenta et al. [1] introduce a reconstruction method based on three-dimensional Voronoi diagram and Delaunay triangulation. In their algorithm, a set of triangles is generated based on the sam-

ple points. Other approaches use interpolating surfaces. Alexa et al. [2] demonstrate the idea of computing the Voronoi diagram on the moving least squares(MLS) surface and adding new points at vertices of this diagram. Likewise, implicit models have also been extensively used. Apart from classical Poisson, Wavelet and radial basis functions [3], complex fitting strategies such as edge-aware point set resampling (EAR) [9] have been explored.

Such methods are effective at reconstruction when the point clouds are sufficiently dense. However, if the point clouds are so sparse that key shape structures are missing or incomplete, these methods are not likely recover the lacking details as the smoothness between the sample points is usually assumed for cost function minimization or regularization.

Recently, data-driven methods have been used toward reconstruction from point clouds. Remil et al. [4] present a new approach that learns exemplar priors from model patches. Then these nearest neighbor shape priors from the learned library can be acquired for each local subset of a given point set. After an appropriate deformation and assembly of the chosen priors, models from the same category as the priors can be reconstructed. Yu et al. [5] develop neural network called

---

PU-Net, to upsample an input point cloud. The network learns point patches extracted/cropped from the point clouds. A joint loss function is utilized in the training process which constrains the upsampled points to be located on the objective surface and distributed uniformly. To the best of our knowledge, PU-Net is the only current data-driven approach to point cloud upsampling. However, the patch-based learning algorithm of PU-Net places high demands on the resolution of the initial point cloud.

In this work, we aim to learn an upsampling strategy using the point clouds of entire objects rather than patches of individual objects. Specifically, we explore how the information contained in the objects belonging to the same as well categories impact the upsampling success. As an example of operating on the point clouds of full objects, Qi et al. [6] developed a deep learning architecture called PointNet that learns features of point clouds tailored for classification and segmentation. Later, they introduce a hierarchical feature learning neural network named PointNet++ [7] capable of extracting both global and local geometric features, with very compelling results.

Achlioptas et al. [8] propose a generative point net model based on PointNet. This generative model is designed to capture the latent generative features of the point clouds used for training using an encoder-decoder architecture. In their recent work, they mention shape completion as one of the potential applictions for their network. Nevertheless, there is no further exploration in upsampling conditions and the categories of objects.

In this work, we build on and extend Achlioptas et al.'s to deploy an upsamling method designed for different object categories and different upsampling amplification factors (AF). Furthermore, we study the attributes of the input clouds that lead to the most accurate upsampled point clouds. Finally, we expand the encoded input point information to incorporate the vertex normals obtained from the original mesh files and evaluate its influence on the reconstruction performance. In our experiments, models from seven categories in ShapeNet-Core [10] are utilized in the training and testing process. The results reveal that data-driven upsampling of sparse point clouds can indeed benefit significantly from categorical class information and more over, the richness in the data (as obtained through multi-class training) results in high-quality upsampled models for a variety of object categories.

The key contributions of our work are as follows:

- We propose a deep learning algorithm for learning point cloud upsampling using entire object models
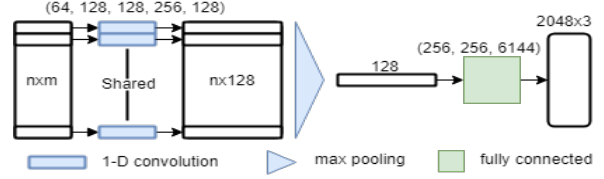

Figure 1: **Network Architecture.** The output dimension of each layer is displayed above layer blocks. 1-D convolutional layers are followed by ReLU and batch-normalization layers. Two ReLU layers are between the fc layers. Reshaping is employed in the last layer.

(rather than patches) as input;

- We demonstrate the effect of input point distribution on upsampling quality;

- We demonstrate the performance of our approach with diverse amplification factors and the flexibility of our algorithm with single and multiple category training scenarios.

2. Technical Approach

2.1. Network Architecture

The neural network, depicted in Figure1, produces a dense point cloud by taking as input a sparse point cloud of an object. The input is a $N \times M$ matrix where N is the number of input points and M is the input dimension of one point, which is either 3 or 6 with normal vectors. The encoder is composed of 1-Dimensional (1D) convolutional layers with filter size 1, followed by batch normalization [15] layer and a ReLU [14]. In each layer, the weights and bias of convolutions are shared among all the points. In the last layer of the encoder, maxpooling is applied along the channel dimension to produce a latent feature vector with a predefined size. Then the feature vector is passed through three fully connected (fc) layers with two ReLU layers in between to complete the reconstruction.

2.2. Loss function

Chamfer distance (CD) [12] and Earth Mover's distance (EMD) [13] are two loss functions most commonly used in training deep neural networks pertinent to point clouds.

The Chamfer distance is defined as

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|^2,$$

where subsets $S_1, S_2 \subseteq R^3$ are two point clouds. Chamfer distance is designed to measure the similarity of two
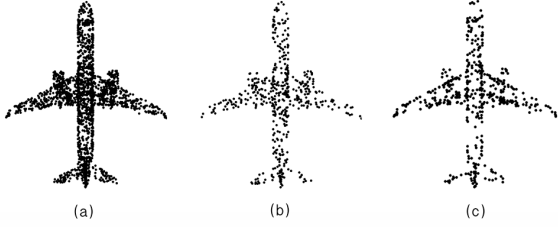
Figure 2: Example results of the two subsampling strategies for the input point clouds. (a) Ground truth point cloud. (b) Uniform subsampling. (c) Curvature-based subsampling.

point clouds. The main idea is to find the nearest neighbour for each point in the other point cloud (and vice versa) and sum the squared distances.

The Earth Mover's distance is defined as

$$d_{EMD}(S_1, S_2) = \min_{\phi:S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|,$$

where ϕ is a bijection of equal size subsets $S_1, S_2 \subseteq R^3$. There exists a unique and invariant optimal bijection ϕ for the point pairs in two sets to move to each other. EMD is a measure of the distance of two point clouds based on the unique bijection ϕ.

After comparison, we adopt the Chamfer distance as our loss function primarily due to its simplicity and due to its better reconstruction qualities reported by P. Achlioptas et al. [8], who study the performance of shape completion when using CD or EMD as loss functions respectively. Chamfer distance achieves much higher accuracy with little loss in coverage in all evaluated categories of objects. The accuracy is defined as the fraction of predicted points that appear in a small range from points in the ground truth. The coverage represents the fraction of points in the ground truth which can be represented by the predicted points. Further details are provided in Appendix A.

3. Experiments

3.1. Dataset and Implementation

ShapeNetCore is a large-scale 3D CAD Dataset collected and processed by Chang et al. [10]. The dataset consists of 55 shape categories. The orientations of the mesh files are aligned. Also, the models are size-normalized by the longest dimension. We select seven categories from ShapeNetCore as our training data: cars, airplanes, boats, benches, chairs, lamps and tables. These seven categories each have more than 1,000 models. The models from these categories ranged from simple to complex, thereby allowing a performance test

of our approach. After selecting a balanced training model set, we divided the data of each category with train/validation/test sets of 85%-5%-10% split. Furthermore, we fix our test cases for all evaluation processes. For learning, we train our neural network for approximately 2,000 epochs using the Adam optimizer [11] with a learning rate of 0.0005 and a batch size of 50. For each category, the training ranges from 40 to 100 minutes using a NVIDIA GeForce GTX 1080 Ti GPU.

3.2. Data Pre-processing

To prepare the point cloud data, for each object, we uniformly sample a point cloud on the original mesh polygons with 2,048 points (larger polygons getting proportionally more samples). Note that the 2,048 points on these models was selected to strike a balance between upsampled model complexity and computational efficiency for the parametric studies discussed in this study. With the insights arrived at with the study, the network architecture can be altered to change the target number of points in the upsampled models. In the subsequent experiments, we take these point clouds as our high resolution ground-truth data. To study the influence of point distributions in the sparse, input point clouds, we downsample each point cloud to 256, 512 and 1,024 points using two approaches: Uniform subsampling (U) or curvature-based subsampling (CB). These subsampled point clouds serve as the input models that our approach aims to upsample. The insight we attain at the end of the study allows a determination of which of the two subsampling approaches (U versus CB) is more suitable for creating accurate upsampled models.

In Figure 3, we show the results of the uniform and curvature-based subsampling respectively. Monte-Carlo random sample method is used to obtain uniformly distributed points. For curvature sensitive sampling, we compute the normal curvature for each vertex in the original mesh. When 2,048 points are sampled, each point is assigned a curvature, which is the linear interpolation of curvatures of the three vertices from the same polygons. Then, input points are sampled from a distribution proportional to these curvatures.

3.3. Experiment Design

Single category training and inner-class evaluation To demonstrate the effectiveness of our network, we conduct experiments under various upsampling amplification factors including 2, 4 and 8. For each amplification factor, we input uniformly distributed points and points sampled by curvature-based sampling method respectively. Moreover, we develop another six cases
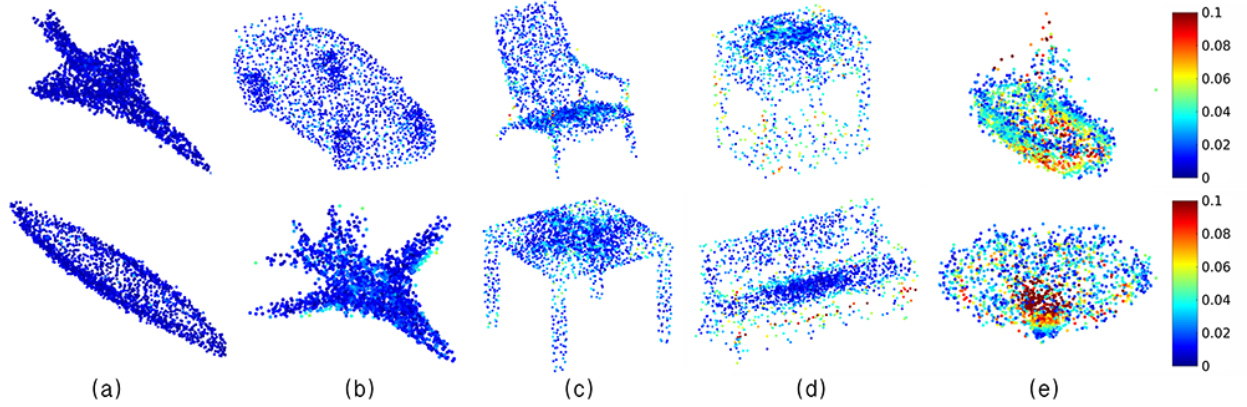
Figure 3: Sample reconstructions from our algorithm with different average test Chamfer loss (ATCL). (a) ATCL= 0.1×10^{-3}. (b) ATCL= 0.5×10^{-3} (c) ATCL= 1.0×10^{-3} (d) ATCL= 2.0×10^{-3} (e) ATCL= 5.0×10^{-3}. The color means the distance between a point in the upsampled point cloud and its nearest point in the ground truth point cloud.

where normal information of the points is taken as additional input. Overall, twelve cases are created for each category where the best case will be picked for further experiments.

Single category training and inter-class evaluation The results of this experiment demonstrate the ability to perform upsampling in previously unseen categories. Models are trained by only one input category and evaluated on the remaining six categories. We apply the condition under which the network models perform best in the first experiment.

Multi-category training and evaluation We randomly pick 1,000 models from each category and train our network on the extracted 7,000 point clouds. Then the network is evaluated on the test cases from each category separately. This experiment aims to study the performance of our network when trained with all available object categories, but tested on models that were previously unseen (though still belonging to one of the seven object categories).

4. Results and Discussions

4.1. Inner-class evaluation

As stated in section 3.3, we conduct parametric studies with twelve different cases to evaluate the upsampling performance of our algorithm. The resulting average test Chamfer losses for each category are shown in Table 1. The best performances for a given amplification factor of each category is shown in bold. We report the accuracy and coverage values as well in Appendix A, which follow a pattern similar to that shown in Table 1.

Among all the best cases, the largest test Chamfer loss is less than 1.7×10^{-3} (minimum 0.285×10^{-3}). When the amplification factor is doubled, the largest increase in Chamfer loss is around 8%. Based on Figure 3, the visualizations of reconstruction quality across different Chamfer loss values, we conclude that overall our upsampling algorithm performs across different objects and multiple amplification factors. Further results are shown in Appendix C.

An interesting observation is that input point clouds that are uniform outperform point clouds that have curvature-based samples in all cases where the amplification factor is 4 or 8. But in four of the seven categories, curvature-based sampled point clouds result in better reconstruction quality at $AF = 2$. Since Qi at el. [6] mention that the points contributing more to the object features usually lie around the edges and outlines of the object, we originally expected points clouds subsampled based on curvature to be more effective at accurate upsampling.

A possible reason is that the input points congregate too much on the edges so that the resulting point cloud is nonuniform. This nonuniformity may decrease the error between the upsampled point cloud and the ground truth since features like sharp edges or high curvature regions require denser point distributions for the features to be faithfully captured. Meanwhile, it inevitably gives rise to the error distance from ground truth to prediction. Because the total number of points is fixed, there must be less points in the flat regions. In cases where the amplification factor is small, the side effect of this nonuniformity is not distinctly revealed since the total number of points is still adequate to cover the flat

Table 1: Evaluation of networks trained on seven categories respectively under twelve conditions. The lowest test Chamfer loss for each amplification factor is marked in bold.

AF	2				4				8			
Sample	U		CB		U		CB		U		CB	
Normal	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Category	Test Chamfer Loss ($\times 10^{-3}$)											
Airplane	**0.285**	0.301	0.289	0.311	**0.294**	0.316	0.313	0.343	**0.319**	0.340	0.356	0.402
Bench	0.814	1.154	**0.771**	2.480	**0.822**	1.089	0.940	1.297	**0.865**	1.273	0.996	1.391
Boat	0.923	1.325	**0.899**	1.410	**0.971**	1.359	1.008	1.449	**1.000**	1.420	1.043	1.206
Car	**0.720**	0.751	0.721	0.738	**0.729**	0.763	0.767	0.806	**0.755**	0.794	0.816	0.866
Chair	**1.322**	1.387	1.335	1.367	**1.353**	1.388	1.792	1.868	**1.414**	1.462	1.933	2.040
Lamp	1.530	2.039	**1.528**	2.161	**1.571**	2.028	1.657	2.163	**1.635**	2.254	1.824	2.665
Table	1.185	1.196	**1.181**	1.196	**1.194**	1.219	1.389	1.418	**1.231**	1.258	1.549	1.593

regions. As AF becomes larger, the unbalanced point distribution eventually results in a dominant threat to the upsampling performance. To verify our hypothesis explained above, a hybrid sampling method with various combining ratios is introduced in section 4.3. However, as the results demonstrate, normal information does not help to improve the upsampling quality in all the cases.

4.2. Inter-class and multi-class evaluation

To further explore the performance of our approach with single-category and multi-category training, we conduct a case study with $AF = 8$, primarily because the largest variance among categories can be observed in Table 1 under this condition. Additionally, based on the results of Table 1, all the input point clouds are sampled from uniform distributions without normal information. Finally, each trained network is evaluated on test cases from the other six categories respectively. The resulting test Chamfer losses are displayed in Table 2.

Expectedly, the average loss increases because the training does not see model features in the other categories used in evaluation. Nonetheless, evaluation categories that are contextually proximate to the training categories interestingly tend to perform relatively better in the inter-class upsampling (car vs. boat, chair vs.

bench etc.). Surprisingly, the network trained on chairs outperforms the network trained on benches among all the six evaluation categories. In fact, the models in the bench category had only 5 major subtypes that were largely consistent in style, whereas the chair database had 23 different subtypes (arm chairs, folding chairs, recliners or even wheel chairs). The richness in the chair models results in the least average Chamfer loss (4.877×10^{-3}). Therefore, we can conclude that greater abundance in training sets can improve the generality of network model.

Finally, we trained a single network that uses all seven categories (balanced) for training. Table 3 shows the performance of this network. Compared with the results in Table 2, this network outperforms all the single-category training networks as it learns a richer set of latent features emanating from different categories. Evaluation Chamfer losses on the bench and boat models become even less than those in Table 1. In addition, the average test loss rises merely from 1.031 to 1.205 when employing this multi-category training.

4.3. Further discussion

Finally, we test a hybrid subsampling method designed to provide a potentially superior input point

Table 2: Inter-class evaluation of the networks trained on single category and evaluated on another category. Uniformly distributed points are used during training as input point clouds. $AF = 8$.

Training Category	Airplane	Bench	Boat	Car	Chair	Lamp	Table
Evaluation Category	Test Chamfer Loss ($\times 10^{-3}$)						
Airplane	\	24.321	102.293	23.312	9.943	3.530	11.599
Bench	77.826	\	35.462	44.513	1.519	6.089	1.649
Boat	3.953	11.938	\	4.273	4.933	2.067	10.881
Car	5.152	5.406	1.543	\	2.601	3.163	4.418
Chair	30.484	5.982	20.687	17.633	\	11.387	4.754
Lamp	12.609	35.19	18.575	56.709	8.846	\	12.545
Table	72.059	5.909	67.493	47.852	4.883	17.199	\

Table 3: Evaluation of the network trained on a balanced training set involving all seven categories. The upsampling condition is the same as Table 1.

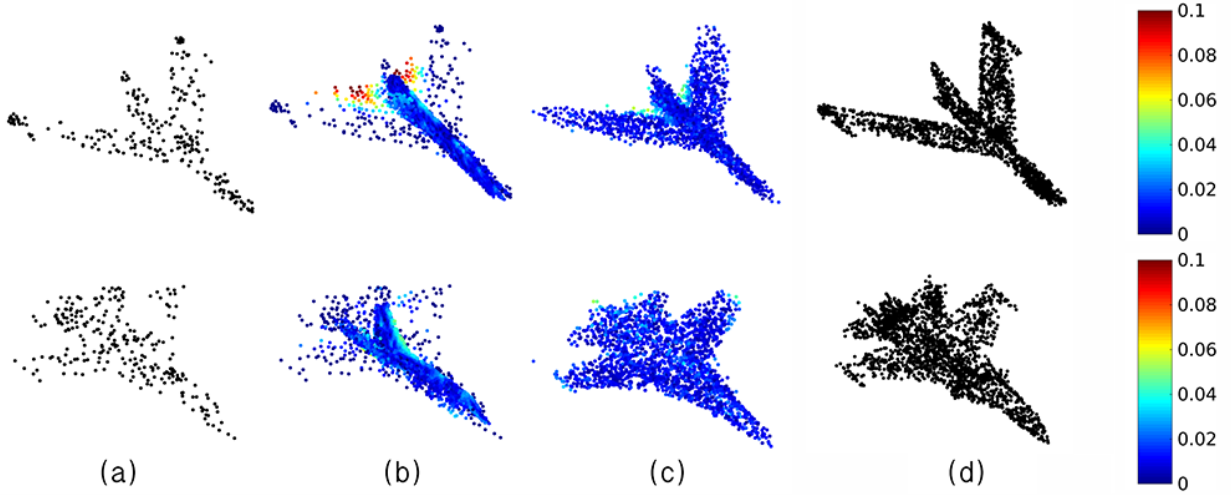| Evaluation Category	Airplane	Bench	Boat	Car	Chair	Lamp	Table
Test Chamfer Loss ($\times 10^{-3}$)	0.529	0.825	0.892	0.854	1.807	1.888	1.644

Figure 4: Comparison between our algorithm and EAR method. (a) Input point cloud. (b) Outcome from EAR. (c) Outcome from our algorithm. (d) Ground truth. The color means the distance between a point in the upsampled point cloud and its nearest point in the ground truth point cloud.

cloud for upsampling. For this, we introduce a new parameter, $\alpha \in [0, 1]$, that represents the fraction of points subsampled using the curvature-based strategy relative to all subsampled points (the rest being the uniformly subsampled points). We create 11 groups of input point clouds from the airplane category with $AF = 8$ and an
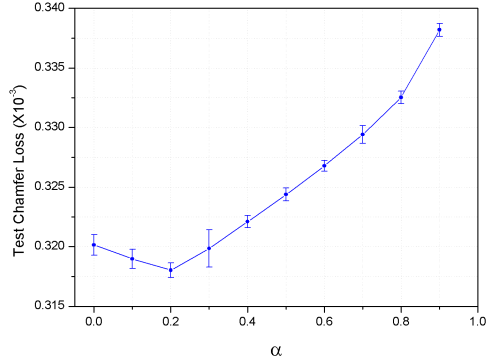
Figure 5: Test Chamfer loss for our trained networks as a function of α.

increment of 0.1 in α each time. The test cases are identical to those we use in Table 1. Figure 5 shows the relationship between α and the corresponding test Chamfer loss. As shown, when $\alpha = 0.1$ or $\alpha = 0.2$, both provide higher upsampling quality over the uniformly sampled points ($\alpha = 0$). As seen, with approximately 20% of the subsampled points coming from the curvature-based approach, the average Chamfer loss is minimized. This implies that there exists a trade-off between using purely curvature-based versus purely uniform subsampled points, with the hybrid approach providing a more desirable outcome. Note that as α increases, the input point cloud becomes much less uniform, and eventually, the improvement resulting from the addition of edge and feature rich regions cannot make up for the lack of points in the flat (low curvature) regions. The accuracy and coverage values reported in Appendix A further verify this inference.

As a comparison with the model we obtained using this hybrid subsampling model, we utilize EAR [9], a state-of-the-art method for optimization-based point cloud upsampling, toward the same task. Figure 4 shows that our algorithm provides more precise upsampling results over the EAR method. Since the number of points is small, the EAR method only finds only some of the major edges and uses most of the points around those regions.

5. Conclusions

This work presents a deep learning algorithm aiming at upsampling a sparse point cloud with a prescribed amplification factor determined by the user. Instead of using human defined priors or heuristics, we exploit the deep networks' ability to extract the latent features for upsampling for various object categories. Then these latent features assist the point cloud upsampling so that a common global feature set learned from a single or a multitude of object categories can be naturally utilized.

We successively explore the effect of two different distributions for input point cloud sampling. Based on the outcomes, a further parametric study on the hybrid sampling ratio for points produced by these two sampling strategies is conducted to identify the benefit of using feature-sensitive points for upsampling. As this ratio increases, points sampled on the high-curvature regions of an object is capable of better capturing the critical feature-rich regions. Meanwhile, a reduction of the point density in areas far from the feature regions causes a scarcity of points in these areas during upsampling, which expectedly decrease the accuracy in the current loss function. Nonetheless, in real applications, fewer points in the flat, featureless regions may not be a severe issue, since by and large most current surface reconstruction methods assume smooth surfaces between the sampled points. As such, our future work will explore alternative metrics to assess the resulting upsampled point cloud so as to enable non-uniform but high quality upsampled models.

6. Limitations and Future Work

Our current model is trained on point clouds sampled from ShapeNetCore. An inspection of the available models reveals that parts of the meshes have geometric shortcomings such as open disconnected polygons or double-sided surfaces. These flaws lead to incorrect normal calculations, which is a potential reason why the normal information did not contribute positively to the upsampling quality. An alternative way is to compute normal vectors using the available vertex coordinates through local patch fitting, which is to be explored in the future.

Another characteristic of our approach is that while it performs well at reconstructing the global features that are frequently occurred in the category, the upsampling quality diminishes for features that are rarely encountered during training. To address this problem, we plan to investigate the possibility of replenishing our network with latent features learned at the local, patch scale.

Finally, our tests thus far have concentrated on models of man-made, engineered objects. Whether the same approach generalizes well for object categories consisting of natural or biological models is the subject of future work.

Appendix A. Accuracy and Coverage

We use the same point cloud upsampling metrics introduced in [8] to illustrate the performance of our algorithm. That is (a) Accuracy: which is the fraction of the predicted points that are within a given radius (ρ) from any point in the ground truth point cloud and (b) Coverage: which is the fraction of the ground-truth points that are within ρ from any predicted point. Referring to Table A.5 and Table A.6, as a supplement to Table 1, the accuracy and coverage values are reported with $\rho = 0.03$.

We explain the principle behind Figure 5 in section 4.2. Here, the accuracy and coverage values are reported in Table A.4. Since accuracy and coverage are both high (98.5%) and close in both cases when $\rho = 0.03$, we use $\rho = 0.015$ in this case.

Appendix B. Benches vs Chairs

To demonstrate the variance in the chair and bench dataset, we randomly select 10 models in each category. The textured models are shown in Figure B.6 and Figure B.7. Larger shape variation in the chair models can be observed.

Appendix C. Upsampling Results

Further upsampling results are shown in Figure C.8, C.9, C.10, C.11 and C.12. In all cases, the left column is the input sparse point cloud, the middle column in our output, and the right column is the ground truth.

Details for each case are listed in the figure caption. (U: Uniform sampling; CB; Curvature-based sampling; ST: Single-category training; MT: Multi-category training. In all cases, we don't use normal information.)

References

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. Proceedings of SIGGRAPH 98, pages 415422, July 1998.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. IEEE Trans. Vis. & Comp. Graphics, 9 (1):315, 2003.

[3] Berger, M., Levine, J.A., Nonato, L.G., Taubin, G., & Silva, C.T. A benchmark for surface reconstruction. ACM Trans. Graph., 32, 20:1-20:17. 2013.

[4] Oussama Remil, Qian Xie, Xingyu Xie, Kai Xu, Jun Wang. Surface Reconstruction with Data-driven Exemplar Priors. arXiv:1701.03230. (2017).

[5] Lequan Yu, Xianzhi Li, Chi Wing Fu, Daniel Cohen, Pheng Ann Heng. PU-Net: Point Cloud Upsampling Network. arXiv:1801.06761. (2018).

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++:Deep hierarchical feature learning on point sets in a metric space. In Int. Conf. on Neural Information Processing Systems (NIPS), 2017.

[8] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. arXiv:1707.02392, 2017.

[9] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang.Edge-aware point set resampling. ACM Trans. Graph. 32, 1, Article 9 (February 2013), 12 pages.2013.

[10] Chang, Angel X. and Funkhouser, Thomas and Guibas, Leonidas and Hanrahan, Pat and Huang, Qixing and Li, Zimo and Savarese, Silvio and Savva, Manolis and Song, Shuran and Su, Hao and Xiao, Jianxiong and Yi, Li and Yu, Fisher. ShapeNet: An Information-Rich 3D Model Repository. 2015.

[11] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In Int. Conf. on Learning Representations (ICLR) ,2015.

[12] Fan, H., Su, H., and Guibas, L. J. A point set generation network for 3d object reconstruction from a single image. CoRR, abs/1612.00603, 2016.

[13] Rubner, Y., Tomasi, C., and Guibas, L. J. The earth movers distance as a metric for image retrieval. IJCV, 2000.

[14] Nair, V. and Hinton, G. E. Rectied linear units improve restricted boltzmann machines. In ICML, 2010.

[15] Ioffe, S.andSzegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

Table A.4: Results of accuracy and coverage for networks as a function of α.

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Accuracy (%)	80.25	80.34	80.34	80.24	80.21	79.96	79.78	79.69	79.65	79.55	79.12
Coverage (%)	83.05	82.92	83.03	82.81	83.02	82.89	82.53	82.55	82.45	82.35	81.55

Figure B.6: Ten randomly selected benches in the dataset.

Figure B.7: Ten randomly selected chairs in the dataset.

Table A.5: Accuracy of networks trained on seven categories respectively under twelve conditions.

AF	2				4				8			
Sample	U		CB		U		CB		U		CB	
Normal	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Category	Accuracy (%)											
Airplane	98.94	98.85	98.94	98.69	98.83	98.6	98.62	98.26	98.48	98.19	98.6	97.45
Bench	92.53	89.36	92.8	66.61	92.57	89.78	91.98	88.71	92.1	89.03	91.47	87.73
Boat	90.1	82.3	89.79	81.71	89.62	82.76	89.09	81.47	89.31	82.16	87.35	79.98
Car	91.94	91.45	91.91	91.77	91.83	91.24	91.18	90.45	91.39	90.56	90.53	89.64
Chair	80.17	79.11	80.02	79.63	79.7	78.96	74.56	73.6	78.98	78.33	72.91	70.94
Lamp	77.41	71.15	77.47	70.33	76.65	70.96	75.95	69.79	75.63	68.06	73.88	64.02
Table	87.15	86.81	86.87	86.58	86.79	86.35	85.13	84.63	86.22	85.87	83.79	83.3

Table A.6: Coverage of networks trained on seven categories respectively under twelve conditions.

| AF	2				4				8			
Sample	U		CB		U		CB		U		CB	
Normal	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Category	Coverage (%)											
Airplane	99.05	98.81	98.93	98.63	98.9	98.58	98.54	98.06	98.43	98	98.58	96.74
Bench	90.3	84.99	90.96	73.82	90.26	85.74	89.31	83.37	89.23	82.81	88.09	82.1
Boat	91.74	84.92	91.89	84.19	91.19	84.24	90.6	82.65	90.74	83.58	89.27	81.78
Car	94.64	93.83	94.38	94.16	94.43	93.53	93.46	92.68	93.94	92.71	92.59	91.42
Chair	81.26	79.71	80.51	80.05	80.48	80.18	76.76	75.52	79.25	78.5	75.02	73.48
Lamp	84.44	76.97	84.72	76.84	84.45	78.28	83.2	76.8	83.78	75.77	81.95	71.94
Table	81.4	81.15	81.39	81.04	81.09	80.72	79.16	78.41	80.64	79.97	77.17	77.07
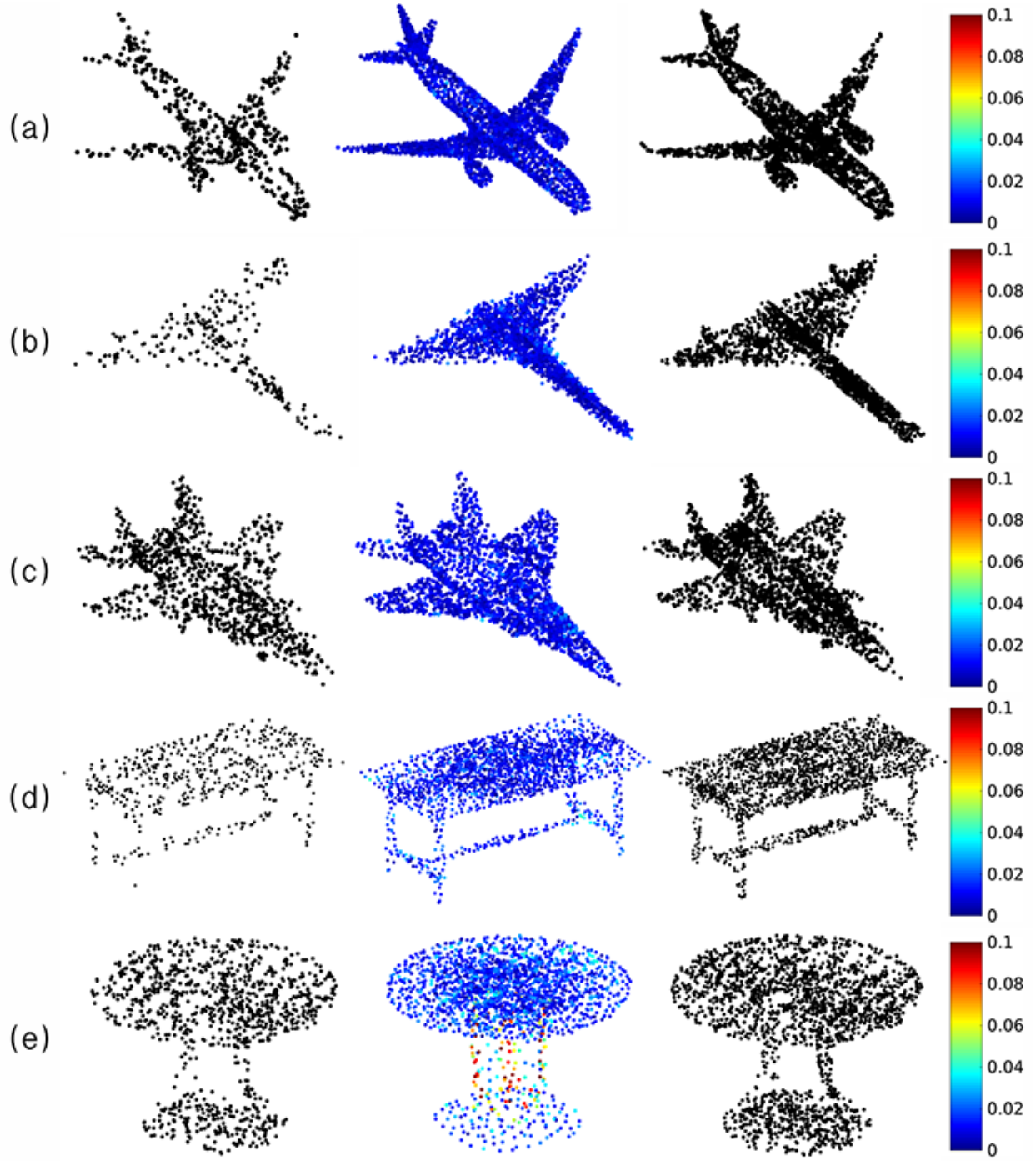
Figure C.8: Sample results (Part I). (a) $AF = 4$, U, ST. (b) $AF = 8$, U, ST. (c) $AF = 2$, U, ST. (d) $AF = 4$, U, ST. (e) $AF = 2$, CB, ST.
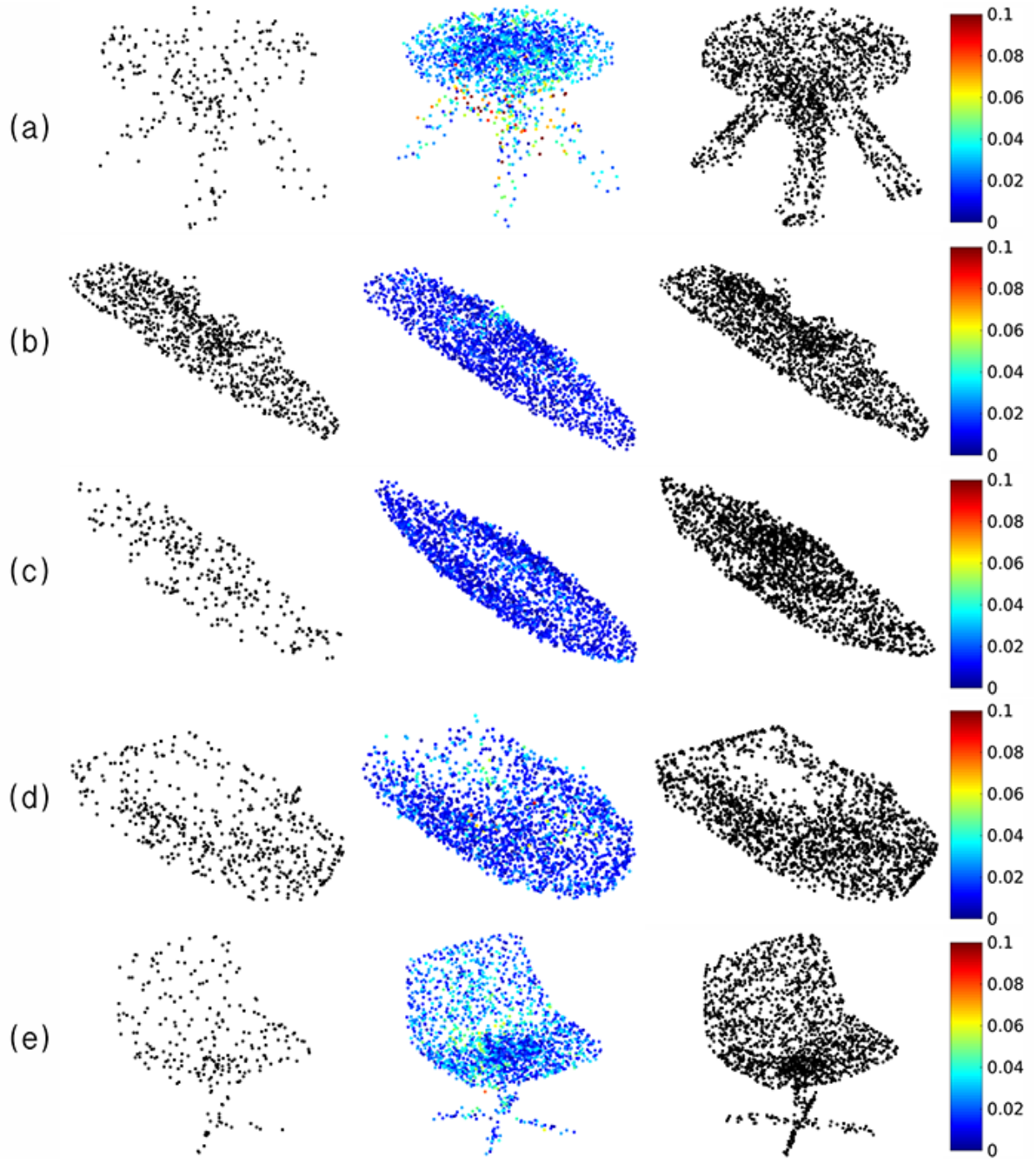
Figure C.9: Sample results (Part II). (a) *AF* = 8, U, ST. (b) *AF* = 2, CB, ST. (c) *AF* = 8, U, ST. (d) *AF* = 4, U, ST. (e) *AF* = 8, U, ST.
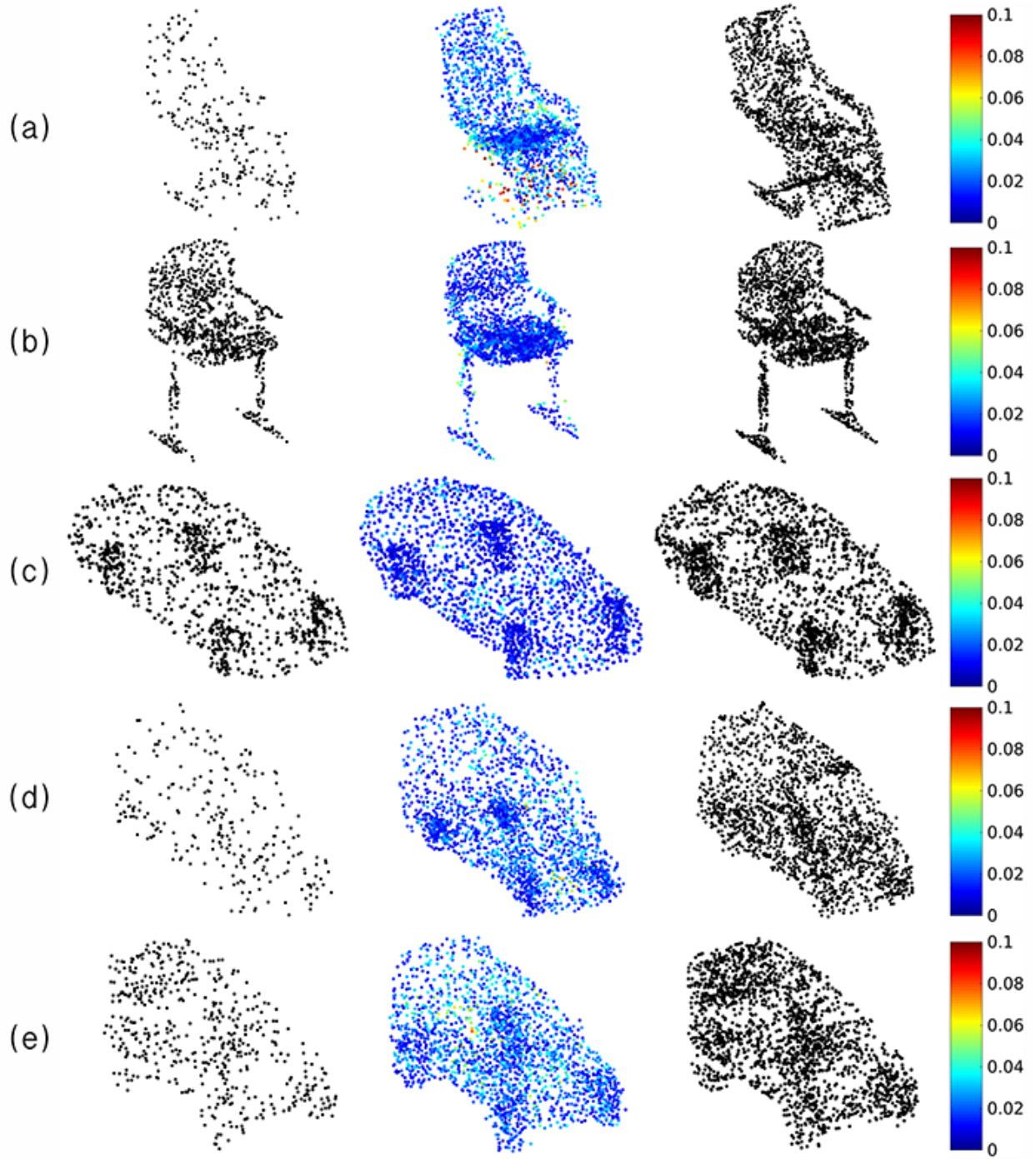
Figure C.10: Sample results (Part III). (a) $AF = 8$, CB, ST. (b) $AF = 2$, U, ST. (c) $AF = 2$, U, ST. (d) $AF = 8$, U, ST. (e) $AF = 4$, U, ST.
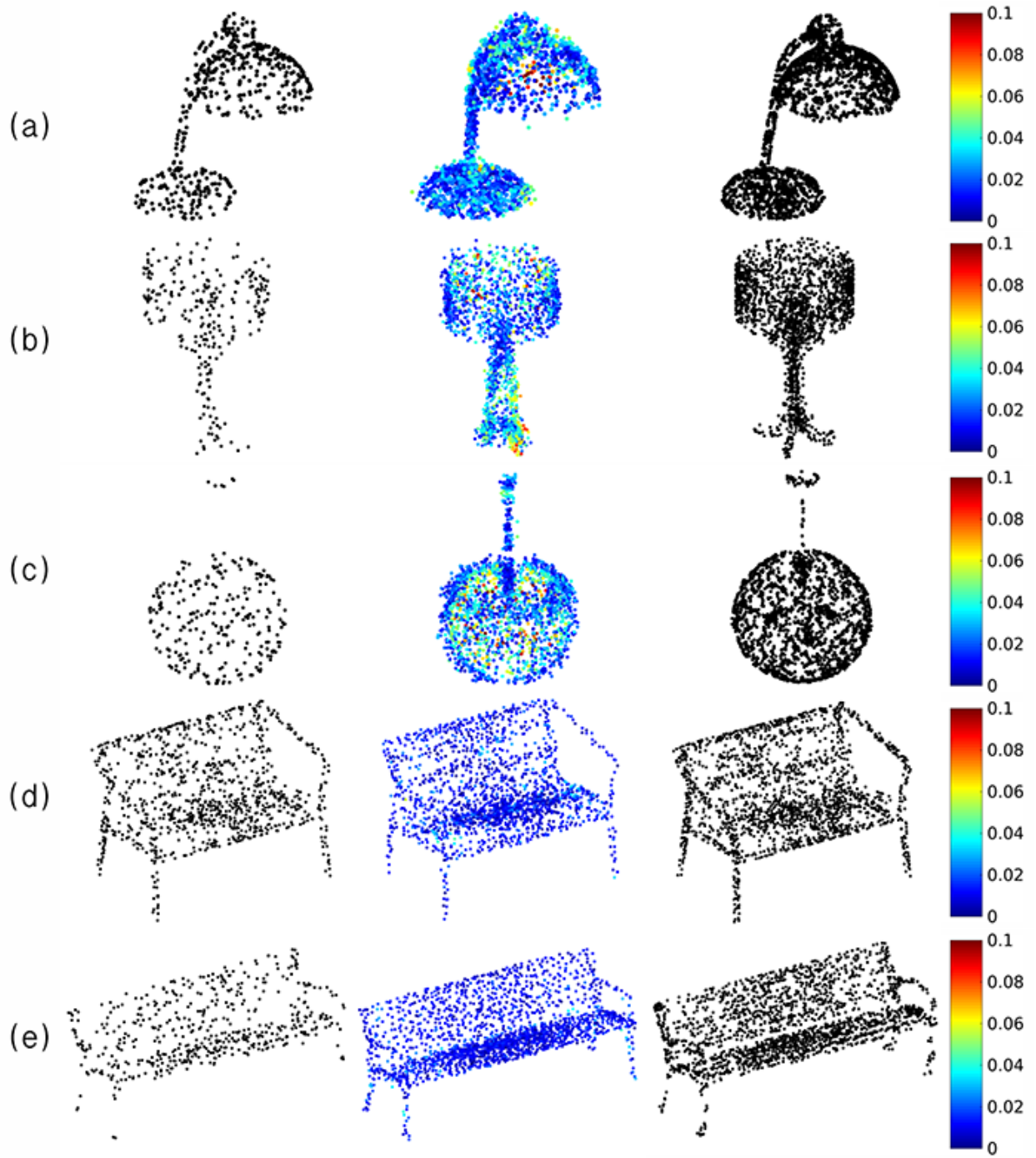
Figure C.11: Sample results (Part IV). (a) $AF = 4$, U, ST. (b) $AF = 8$, U, ST. (c) $AF = 8$, U, ST. (d) $AF = 2$, CB, ST. (e) $AF = 4$, U, ST.
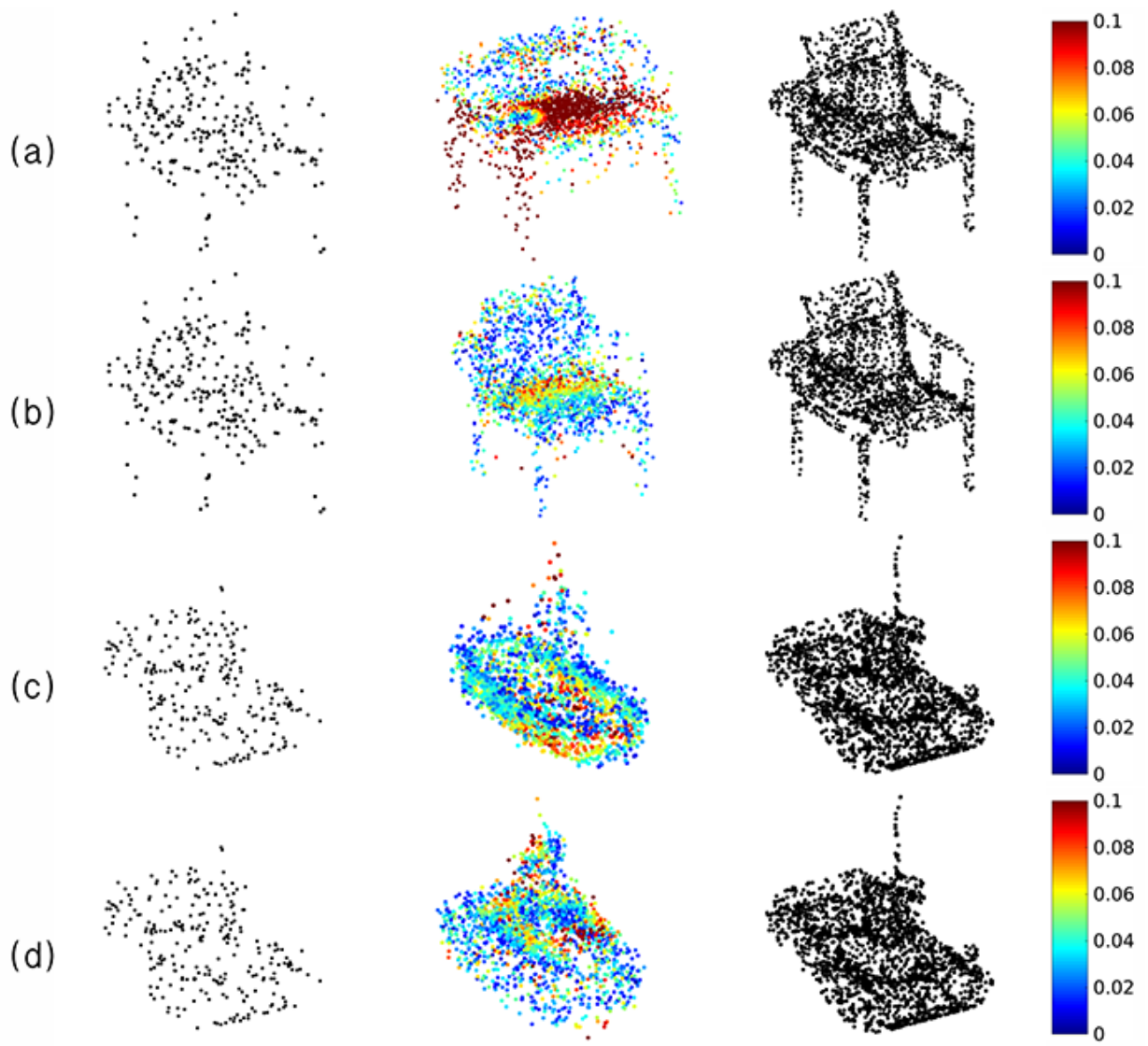
Figure C.12: Sample results (Part V). (a) $AF = 8$, U, ST. (b) $AF = 8$, U, MT. (c) $AF = 8$, U, ST. (d) $AF = 8$, U, MT.