Supplementary Mateirals
SPNR: Generalizable Sparse-Point Neural Rendering

Xuyi Meng[1,2] Jialin Zhang[1,3] Fanbo Xiang[1] Jiayuan Gu[1] Xiaoshuai Zhang [1]
Hao Su[1†]
[1]UC San Diego [2]Nanyang Technological University [3]Tsinghua University

In the supplementary, we provide further experiment details on architecture, training, datasets and baseline implementations. We will also present addtional quantitative and qualitative results on all datasets.

A. Additional Experiment Details

1. Implementations

Architecture We use a multilayer perceptron (MLP) to implement the PointNet, which contains 3 hidden layers with (64, 128, 256) units. Each hidden layer is followed by a batch normalization and a ReLU activation.

The 3D U-Net in our work is deeper than that in MVS-NeRF [3]. In each stage of the encoder, the spatial resolution is reduced by half, and the feature dimension is doubled. The final resolution of the feature volume in the encoder is 1/32 of the input one. The skip connections in the decoder are implemented as addition.

We use stratified sampling similar to NeRF [6] during the rendering process. The first stage is coarse sampling, where 64 points are sampled along each ray uniformly. The second fine sampling stage re-samples 64 points along each ray. In this stage, more points are sampled near the points with larger density in the coarse stage. The points sampled from both stages are combined as the final shading points.

The super-resolution module is slightly different for each dataset. The output resolution of our volumetric rendering is 64×64. For ABO [4], we use a 2D CNN to upscale the coarse image to 128×128. For ShapeNet [2], we first upsample it to 100x100 by bilinear interpolation, followed by a 2D CNN to upscale it to 200x200.

Training We use the adversarial loss with a weight of 1 and L1 reconstruction loss with a weight of 100. We observe that different datasets demand different loss weights for high perceptual quality. For example, on the ShapeNet-chair dataset, we observe cracking patterns at convergence as shown in Figure 1 when the weight of L1 loss is small compared to the adversarial loss. In this case, we specially increase the L1 loss weight for ShapeNet-chair class to 5000, while the trainings on other datasets still remain 100. All the qualitative and quantitative results in the following are based on this setting.

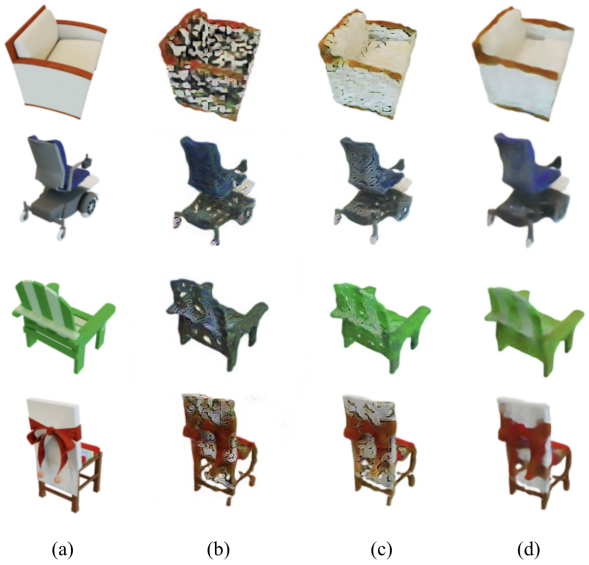

Figure 1. Trade-offs between reconstruction loss and adversary loss. We show reference image in column (a) and results with adversary loss with weight 1 and L1 loss with weights 0, 100, and 5000 in column (b), (c), and (d) respectively. The adversary loss favors sharp image features but tends to also introduce high-frequency artifacts, while L1 loss favors smooth images.

2. Datasets

We present more details about the datasets we used. Our experiments are conducted on two datasets, including Amazon Berkeley Objects(ABO) [4] and three classes of ShapeNet [2].

Amazon Berkeley Objects We use Amazon Berkeley Objects which contains 7953 products with artist-designed high quality 3D meshes. It covers 63 classes and features great diversity of geometry and appearance. Most of the ob-

Figure 2. The example of color augmentation. The original object is on the left and the result after color augmentation is on the right.

jects are furniture with wood or fabric textures. To enhance appearance diversity, we augment each ABO object with 5 variants by transforming its base colors. For each variant, we apply a single random rotation in the RGB space to all base colors. Figure 2 shows a picture frame and an augmented variant. This augmentation increases the color variance while preserving distance between original material colors. There are 16161 training objects (including variants) and 200 test objects (without color augmentation).

ShapeNet-Plane, Car, Chair For the ShapeNet dataset, we uses the car, plane and chair category similar to Points2NeRF [10]. We do not apply color augmentation for ShapeNet. And there are 3627, 1127, 4882 training objects and 406, 128, 105 test objects for plane, car, chair respectively.

Sample Point Cloud and Render Reference Images We uniformly sample point cloud on the surface of each mesh in our datasets. All points are sampled with material properties, including base color, roughness and metallic values. We discarded meshes with additional textures such as transparency textures. All these material properties are read from principled BSDF loaded in Blender.

For reference images, which are used as the ground-truth image in reconstruction losses and "real" data in GAN losses, we render 10 RGB images with white backgrounds from random viewpoints. We use the Blender Cycles renderer and a fixed indoor environment map[1] as lighting. For ABO, the target resolution is 128×128; and for ShapeNet, the target resolution is 200×200 following Points2NeRF [10].

Render Visible Points as Discriminator Condition To ensure the appearance consistency between the rendered image and input point cloud, the discriminator of our pipeline is conditioned on visible input points. Given an image-based discriminator, we choose to represent the condition by an image rasterized from the point cloud (denoted by *condition image*). To be specific, we compute all visible faces given a camera pose via rasterization, and preserve the input points sampled from visible faces only. These visible points are projected onto the *condition image*, and the RGB values at these pixel coordinates are fetched from the reference images.

[1]Asset from ambientCG.com, licensed under the Creative Commons CC0 1.0 Universal License. https://ambientCG.com/a/IndoorHDRI003

3. Baselines

3.1 NPBG

The original NPBG [1] optimizes point-wise neural descriptors per scene, which is not applicable to our setting. Thus, we adapt it to take non-learnable material properties instead of learnable neural descriptors as inputs. We follow the implementation of rasterization and rendering network in [7] [2]. Concretely, the input point cloud is rasterized to a pyramid of 5 images of different spatial resolutions. The rendering network is a 5-stage U-Net with gated convolutions. The rasterization images are concatenated to corresponding stages of the U-Net encoder. The model is supervised by an L1-loss rather than the perceptual loss. Our preliminary results show that the perceptual loss fails to encourage the model to condition on input colors. We train the model on each dataset for 20 epochs. We do not observe significant improvement if the model is trained for more epochs (e.g., 100).

3.2 Point-NeRF

We use the released code base[3] of Point-NeRF [8]. Since per-scene point growing and per-point feature fine-tuning in Point-NeRF are not applicaple to our settings, we only adopt their cross-scene training pipeline, and we use non-learnable material properties as the per-point features. We also replace their ball query and KNN module with our own implementation, as we cannot find a way to effectively increase the ball-query radius in the original code base. We find that increasing the ball-query radius can significantly improve image quality in Point-NeRF, probably due to the sparsity of our point cloud. The only changes we make to the original model are that we increase the number of shading points from 40 to 64, and we use a ball-query radius of 0.1 in the world space. In the training stage, we supervise with color reconstruction loss only to match our settings. We train Point-NeRF for 1M steps with a ray sample size of 1024; further training does not significantly improve the image quality on the training set.

[2]https://github.com/rakhimovv/npbgpp
[3]https://github.com/Xharlie/pointnerf

3.3 Points2NeRF

We use the original code[4] open-sourced by the authors. We are not able to compare with it quantitatively as its training fails to converge on the ABO dataset, and takes too long on each class of ShapeNet, so we use their pretrained models of each class released by the authors. Their pretrained models seem to use a non-standard and inconsistent coordinate system that we could not align with our settings. For qualitative comparison, we use their pretrained models and manually choose views that roughly align with ours, and do inference on our test set using 1024 points as input. Note that they use a different environment map, and thus their qualitative results are also not directly comparable. Even though we use exactly the original code, original pretrained models and original dataset, we still cannot reproduce the result on the chair class of ShapeNet. For quantitative results, we directly report PSNR scores claimed in the original paper of Points2NeRF [10].

3.4 Poisson Surface Reconstruction

For the classic approach Poisson Surface Reconstrucion [5], we use the implementation of Open3D [9]. Since Poisson Surface Reconstruction requires point cloud with normals, we first use Open3D to estimate the normals and then perform the surface reconstruction. We have tried many possible depths of the octree used for the surface reconstruction, but all results are not so satisfying due to the sparsity of point cloud. We choose $\mathrm{depth} = 12$ for the reconstruction, copy material properties from the input points to the output mesh vertices, and the reconstructed meshes are rendered with same setting using Blender Cycles Renderer.

B. Additional Results

1. Comparison on more Datasets

In addition to ABO and ShapeNet airplane, we provide quantitative comparison on 2 more ShapeNet classes, car and chair in Table 1. We provide qualitative comparison of more objects in Figure 4 and 5. We additionally provide qualitative comparison of more than 100 objects for each dataset at the end of the appendix.

2. Depth Results

Our pipeline is also capable of producing depth maps for corresponding RGB images since it uses a 3D volumetric representation. Although no direct supervision is applied to the generated depth maps, the volumetric density inferred through RGB supervision can be used to derive depth maps from a weighted sum of shading point depths along each camera ray, as implemented in the original NeRF [6]. Figure 3 shows that our method generates consistent and clean depth map on most datasets, except for ShapeNet-plane. We speculate that the depth artifacts of ShapeNet-plane is caused by the small occupancy of plane objects in 3D space, as well as the similarity between the white color of planes and the white background. Such similarity could result in ambiguity that poses challenges for our model to tell apart the foreground and background in certain regions, thus resulting in floaters in the rendered depth map.

[4] https://github.com/gmum/points2nerf

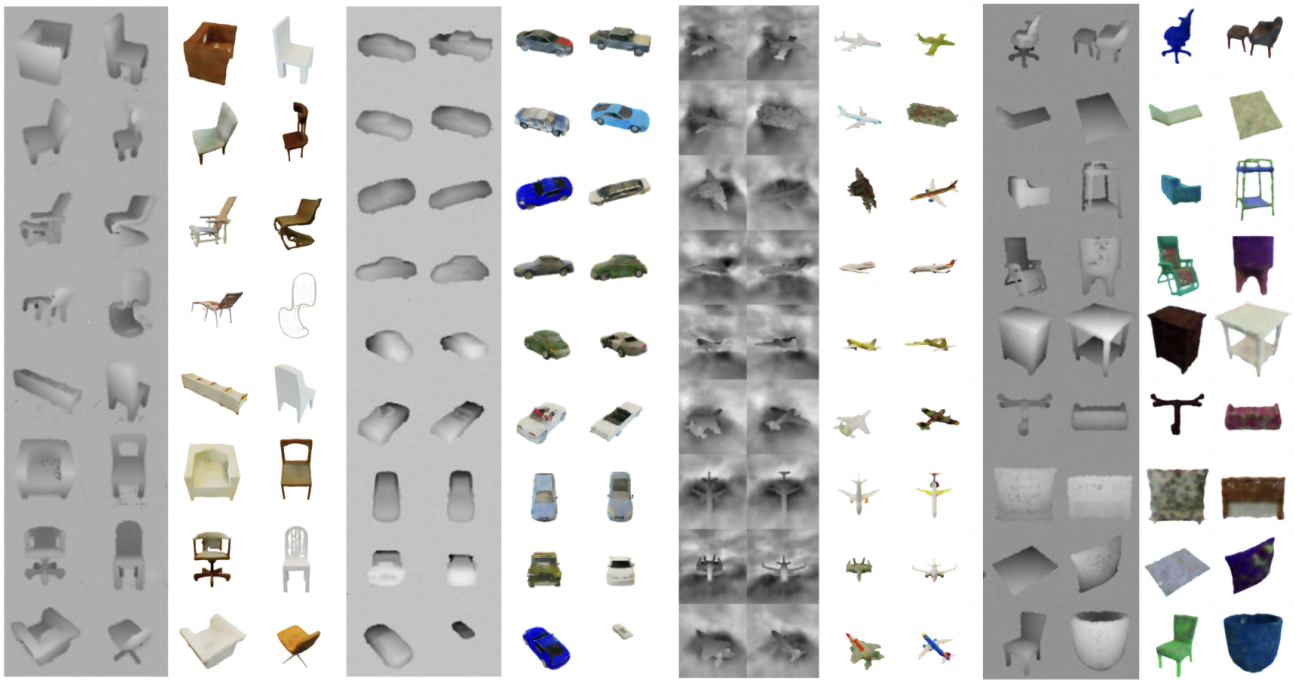

Figure 3. Depth rendering result on different datasets: (a) chair, (b) car, (c) plane, (d) ABO.

	ABO			Car			Airplane			Chair		
Method	PSNR↑	LPIPS↓	cPSNR↑	PSNR	LPIPS	cPSNR	PSNR	LPIPS	cPSNR	PSNR	LPIPS	cPSNR
NPBG [1]	16.7	0.173	13.3	16.3	0.157	8.51	**25.6**	0.0639	13.8	19.2	0.133	13
Point-NeRF [8]	20.5	0.214	15.2	16.6	0.181	9.33	**25.6**	0.0879	**14.5**	18.2	0.193	9.34
Points2NeRF* [10]	-	-	-	20.86	-	-	20.45	-	-	17.17	-	
Ours	**22.3**	**0.110**	**17.0**	**19.9**	**0.0781**	**18.4**	25.4	**0.0442**	14.4	**20.9**	**0.0939**	**16.6**

Table 1. Quantitative comparison. *PSNR values are copied from Points2NeRF paper.

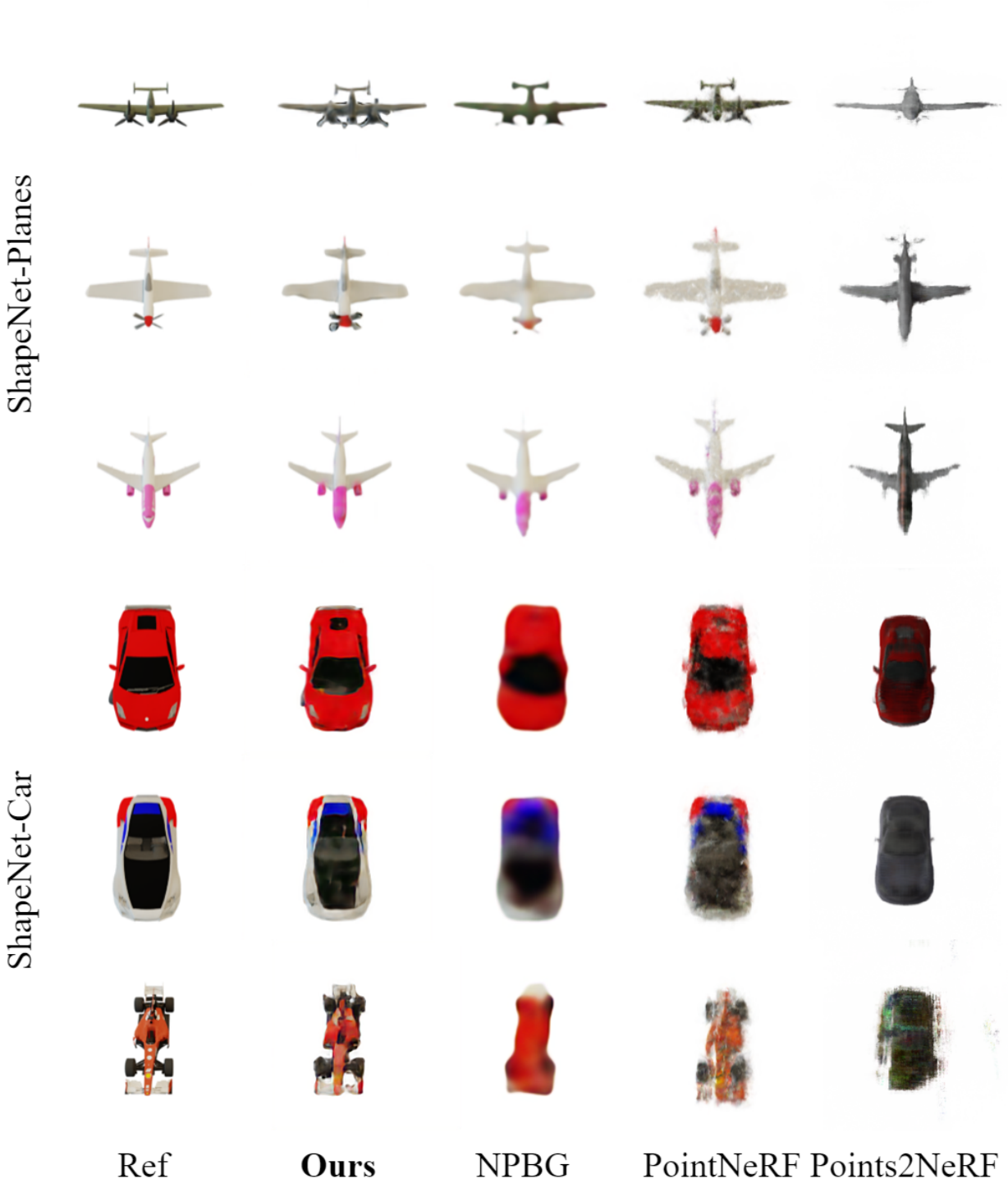

Figure 4. Result on ShapeNet-Planes and ShapeNet-Carss.

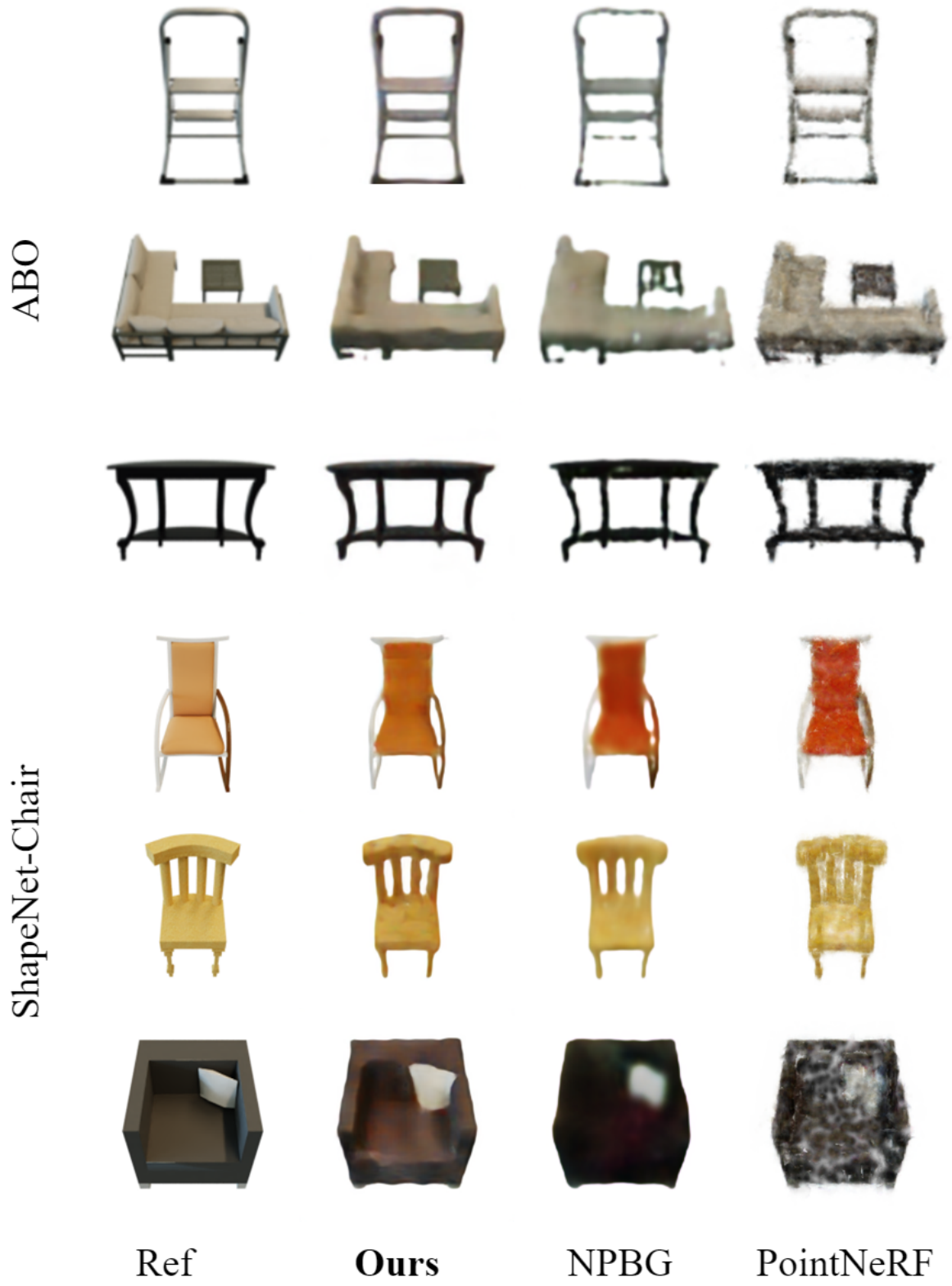

Figure 5. Result on ABO and ShapeNet-Chairs. Points2NeRF fails on ShapeNet-Chairs even we used exactly the original code, original pretrained models and origianl dataset the authors released, and fails to converge on ABO dataset.

Figure 6. Reference images on ABO dataset.

Figure 7. Our method results on ABO dataset.

Figure 8. NPBG results on ABO dataset.

Figure 9. Point-NeRF results on ABO dataset.

Figure 10. Reference images on ShapeNet-Plane.

Figure 11. Our method results on ShapeNet-Plane.

Figure 12. NPBG results on ShapeNet-Plane.

Figure 13. Point-NeRF results on ShapeNet-Plane.

Figure 14. Reference images on ShapeNet-Car.

Figure 15. Our method results on ShapeNet-Car.

Figure 16. NPBG results on ShapeNet-Car.

Figure 17. Point-NeRF results on ShapeNet-Car.

Figure 18. Reference images on ShapeNet-Chair.

Figure 19. Our method results on ShapeNet-Chair.

Figure 20. NPBG results on ShapeNet-Chair.

Figure 21. Point-NeRF results on ShapeNet-Chair.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. June 2019. 2, 4

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[3] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021. 1

[4] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *CVPR*, 2022. 1

[5] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006. 3

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision – ECCV 2020*, pages 405–421. Springer International Publishing, 2020. 1, 3

[7] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15969–15979, June 2022. 2

[8] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022. 2, 4

[9] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 3

[10] D. Zimny, T. Trzciński, and P. Spurek. Points2nerf: Generating neural radiance fields from 3d point cloud. June 2022. 2, 3, 4