SPARF: Large-Scale Learning of 3D Sparse Radiance Fields from Few Input Images
Supplementary Material

Anonymous ICCV submission

Paper ID 24

1. Detailed Formulations

1.1. Sparse Convolutions

Sparse convolutions are a variant of standard convolutions that are used in deep learning. In a sparse convolution, only a subset of the input elements is used in the computation, which allows for more efficient use of computation resources and can improve the performance of the convolutional neural network. To perform a sparse convolution, we first define a set of indices that specify which input elements should be used in the convolution. This set of indices is called the "support" of the convolution. We then use these indices to select the relevant input elements and compute the convolution using these elements. This is typically done by applying a filter to the selected input elements and summing the results to produce the output of the convolution.

In the simplest 1 D case, let $\mathbf{x}$ be the input tensor, $\mathbf{w}$ be the convolutional filter, and c be the support of the convolution (i.e. the set of indices specifying which elements of x should be used in the convolution). The output of the sparse convolution, $\mathbf{y}$, can be computed as: $\mathbf{y} = \mathbf{x}[c] * \mathbf{w}$ where $*$ denotes the convolution operation, and $\mathbf{x}[c]$ is the subset of elements from $\mathbf{x}$ specified by the support c. This equation applies the convolutional filter $\mathbf{w}$ to the selected input elements and sums the results to produce the output of the convolution. For more detailed formulation and implementation of the Sparse convolutions we used in our work, please refer to MinkowskiNetwork [3].

1.2. Q-Gaussian loss sampling

In the space of sparse voxels of high resolution, defining *where* the loss is sampled is difficult, especially if the output topology is unknown. One of the challenges in working with sparse voxels of high resolution (*e.g.* 512) is that training can not involve densifying the voxels to the original resolution (*i.e.* 512^3) due to prohibitive memory requirements. The input/output topologies are not necessarily the same, as the sparse convolutional strides and pruning can alter the sparse voxels' coordinates. This is why the sampling function $\mathcal{S}$ in Eq (**??**) is of utmost importance in guiding the training of SuRFNet. We sample at random coordinates centered at the center of the voxel grid $\mathcal{S}: \ \mathbf{c} \sim \mathcal{Q}\left(\mathcal{N}(\frac{\mathbf{H}}{2}, \frac{\mathbf{H}\sigma^2}{2}\mathbb{I})\right)$, where $\mathbf{H} = (H, H, H)$ is the voxel grid resolution vector, $\mathbb{I}$ is the identity matrix, σ is a hyperparameter determining the spread of the loss, and $\mathcal{Q} : \mathbb{R}^3 \rightarrow \mathbb{Z}^{+3}$ is the quantization-and-cropping function of coordinates that ensure the output coordinates are integers within bounds $\mathbf{c} \in [0, 1, ..., H-1]^3$. We discuss more details about $\mathcal{S}$ and alternative configurations in Section 3.3. Simply put, the Q-Gaussian is 3D normal distribution quantized to integer coordinates to give prior about where the output is expected and where the loss is defined.

2. Detailed Setup

2.1. SPARF Dataset

All the rendered images are of 400×400 resolution with 4 channels (RGB + alpha channel for background). SPARF has three main splits for every 3D shape: training views (400 views), test views (20 views), and an OOD "hard" views (10 views) as can be shown in Figure 13. Regarding the collected SRFs, Plenoxels [4] is used as the base module. The spherical harmonics dimension of the whole SRFs is $d = 4 \times 3 = 12$, while for partial SRFs, it is $d = 1 \times 3 = 3$. Most of the hyperparameters used in optimizing the SRFs are the default ones proposed in the Plenoxels paper [4] (as can be seen in the attached code under Svox2/opt/opt-py). However, the following hyperparameters were engineered in order to scale up the optimization and maintain the quality of the SRFs (as can be seen in Figure 14, and 15). The flickering temporal noise introduced in the 32^3 resolution SRFs is due to the extremely low number of voxels representing the radiance fields while the views are rendered densely from the spiral sequence, hence aliasing occurs.

Running Plenoxels [4] for fewer iterations (3×12K) reduces the time by 30% while maintaining the same PSNR.

SRF Type	Voxewl Resolution	Nb. of Variants	Nb. of SRFs
Partial	32	4×1-view	158,816
		4×3-view	158,816
	128	4×1-view	158,816
		4×3-view	158,816
	512	4×1-view	158,816
		4×1-view	158,816
Whole	32	1×400-view	39,704
	128	1×400-view	39,704
	512	1×400-view	39,704
Total	-	-	1,072,008

Table 1. **SPARF Anatomy**. We show the distribution of the one million SRFs collected in SPARF between multiple resolutions and between whole and partial SRFS.

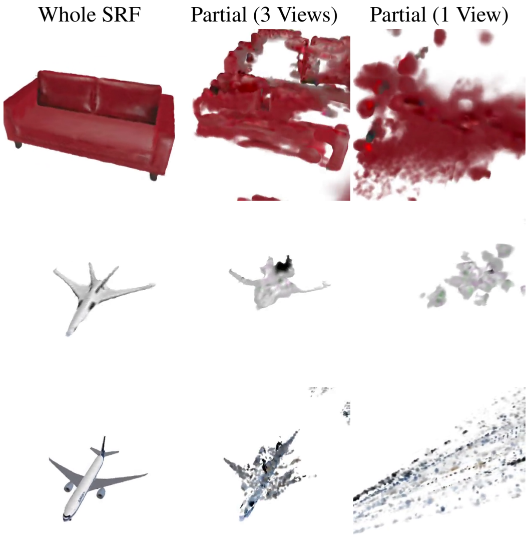

Figure 1. **Whole *vs*. Partial SRFs**. The partial SRFs are used instead of the few images that generated them as input to the learning pipeline to generate the whole SRFs

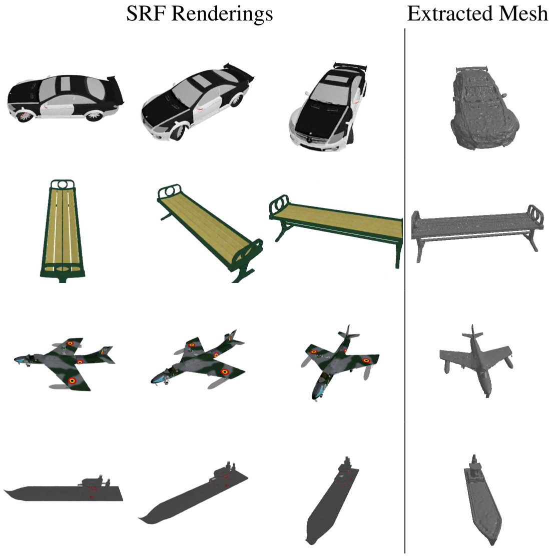

Figure 2. **Extracting 3D Meshes from SRFs**. Since SPARF and SuRFNet live on the 3D voxel's space, extracting the mesh is straightforward with one pass of MarchingCubes [8].

Using 400 views/shapes in SPARF to optimize the SRFs keep the time manageable in optimization (~ 4 minutes for the 512 resolution) while maintaining high PSNR ($\sim$ 30dB). When saving the SRFs, we only save the set of coordinates (integers) and float features (densities and radiance components). The upsampling iteration of Plenoxels is set to $1\times$12K for faster convergence. The distribution of the collected dataset is detailed in Table 1. More examples of the whole *vs*. partial SRFs collected in SPARF can be found in Figure 1. A total of 200K GPU hours are used in the optimization process to collect SPARF. The whole SRFs are easily convertible to high-quality meshes using Marching Cubes [8] as shown in Figure 2. While the SPARF dataset is indeed larrge in total (3.4 TB), its posed-images part is only 360 GB, which makes it manageble for training on other applications that require dense posed images.

2.2. SuRFNet Training

We use a voxel resolution of 128^3 of the SPARF dataset in most of the learning experiments and visualizations in this work, unless otherwise clearly stated. This choice is to reduce the computational cost of training the heavy pipeline and to facilitate the development of proper learning methods on SRFs. The input SRF is normalized with a fixed value of 10,0000 for the density and 10 for the colors, to ensure the distribution lies within -1 to 1. The Q-Gaussian std σ is set to 0.444 (studied more in Section 3). The strides for the SuRFNEt are all set 2, while the network depth $l = 3$ modules. The batch size for training is 14 when A100 GPUs are used and 6 when V100 GPUs are used. The training saturates at 100 epochs. The optimizer used is AdamW [9] with a learning rate of 0.01, a momentum of 0.9, a weight decay of $1e-5$, and a learning rate exponential decay rate of 0.99. The hyperparameters $\lambda_R, \lambda_\alpha, \lambda_\rho$ are all set independently to each class, where a different network is trained on each class separately. Most classes have $\lambda_\alpha = 30.0, \lambda_\rho = 1.0, \lambda_R$. We did not prune the output sparse voxel as this leads to harming performance most of the time and increase the problem of vanishing gradients. The background color of the rendered images $\mathcal{R}_\phi\left(\mathbf{F}(\mathcal{X})\right)$ is masked out from the perceptual loss and the density component α of the output SRF is also not affected by the perceptual loss, as this can cause excessive densities around the object, leading to deteriorating the SRF output perceptuality. During training with the perceptual loss, three randomly selected images from three different

ϕ as used as labels for the three rendered images from the output $\mathcal{R}_{\phi}\left(\mathbf{F}(\mathcal{X})\right)$. The SuRFNet is jointly predicting the density and radiance of spherical harmonics colors, but with different heads. More setup details can be found in the attached code and analyzed further in Section 3. We train a separate model for each class, to maintain high-quality generation of 3D SRFs.

Retraining Baselines. To compare to the preprinted baselines PixelNeRF and VisionNerf (which use 64×64 resolution), we upsample their resolution at inference at test poses while using their pretrained weights of the NMR dataset. The upsampling is using the bicubic sampling of the Pytorch Transforms library. These baselines are used in this works by default unless otherwise specified.

Retraining the methods from scratch on the high resolution 400×400 is computationally expensive. To illustrate the retraining cost, VisionNeRF [7] was originally trained on *16 A100 GPUs for a week* just to converge on 64×64 resolution. SPARF's 400×400 resolution images would need ~ 39 times as much time/compute due to per-pixel sampling. In contrast, our SuRFNet was trained on *a single V100 GPU* for 3 days, which allows for fine-tuning of the learning pipeline for quick convergence. However, for a fair comparison, We retrain PixelNeRF [14] on the 13 categories and report the 3-view and 1-view PSNR results in Table 2. We see that our SuRFNet indeed surpasses this baseline trained on the same SPARF data. The results of retraining are not very different from the pretrained weights (slight improvement) because training on SPARF high-resolution images is unstable using these 2D-based NeRF networks that need a per-pixel sampling, which diverges the training in some cases. The full results are shown in Table 2

3. Additional Analysis

3.1. Shiny Objects

Some of the rendered objects have reflective materials, resulting in distorted optimized radiance fields for these shapes despite using all of the views. We separate these distorted SRFs (only 76 shapes in total) from the SPARF dataset (see Figure 3).

3.2. Effect of Dataset Size

We study the effect of increasing the dataset size (Whole SRFs and Partial SRFs) on the generalization performance of SuRFNet in Figure 5,4. It shows that as the dataset size increase (normalized the number of shapes in each class), the generalization performance increase. This scalability effect underlines the importance of SPARF. However, as can be seen from these two figures, partial SRFs scalability is more important than increasing whole SRFs, which justifies collecting 4 variants per resolution (as detailed in Table 1). This observation aligns with previous generative models in the literature [5, 13, 6]

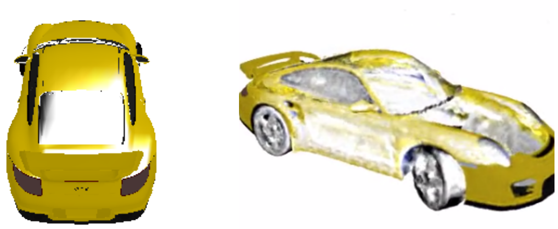

Figure 3. **Shiny Objects Corrupts SRFs:**. Optimizing SRFs on shiny objects with a reflective material (*left*) results in distorted radiance fields (*right*). These distorted SRFs (of 76 shapes in total) were separated from the main classes in SPARF.

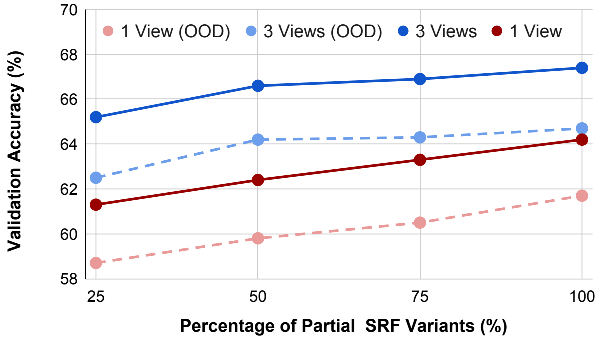

Figure 4. **Scaling-Up Training on SRFs: Partial SRFs**. As the training data (partial SRFs) of radiance fields increase, the generalization improves, as can be seen in the car class here. The 3-view and 1-view metrics are reported with test and OOD metrics.

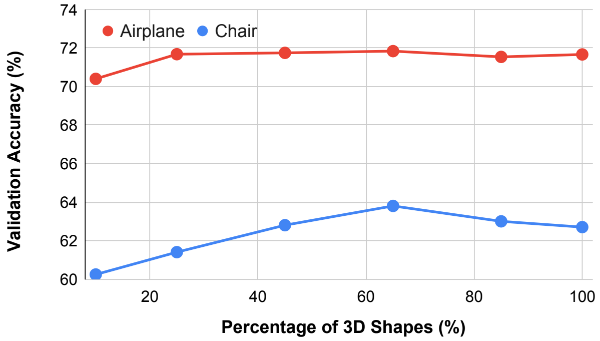

Figure 5. **Scaling-Up Training on SRFs: Whole SRFs**. As the training data of radiance fields increase, the generalization improves across different classes in SPARF.

3.3. Loss Ablation Study

Loss Sampling. We study the effect of the sampling strategy with a different number of input images at test time on the performance of SuRFNet in Table 3. It shows that using a

	SPARF Classes													
Baselines	chair	watercraft	rifle	display	lamp	speaker	cabinet	bench	car	airplane	sofa	table	phone	mean
Plenoxels [4] (1V)	10.1	12.1	12.6	8.7	14.7	8.7	10.9	11.4	7.7	14.0	9.7	10.5	9.5	10.8
Plenoxels [4] (3V)	10.8	13.3	15.6	9.7	16.2	10.1	12.1	12.1	9.0	15.4	11.4	10.8	10.2	12.1
PixelNeRF [14] (1V)	10.8	14.1	14.2	9.0	15.6	9.2	10.5	12.4	10.1	15.7	11.1	10.6	11.1	11.9
PixelNeRF [14] (3V)	11.0	14.1	14.2	9.3	15.7	9.4	10.6	12.7	10.1	15.7	11.3	10.9	11.4	12.0
PixelNerf * (1V)	13.8	12.2	15.0	17.5	19.0	11.2	17.5	13.3	12.2	17.9	11.3	11.7	14.7	14.5
PixelNerf * (3V)	17.5	13.6	16.2	11.7	20	15.6	13.1	17.6	12.1	18.2	16.2	12.1	10.7	15.0
VisionNeRF [7] (1V)	16.5	18.4	18.5	15.1	19.3	13.2	16.1	16.3	13.8	21.8	15.1	14.8	14.0	16.4
SuRFNet (ours) (1V)	15.7	15.5	19.1	14.1	18.5	14.5	18.7	15.6	18.1	20.3	16.3	14.1	17.4	16.8
SuRFNet (ours) (3V)	**18.6**	**20.7**	**20.9**	**17.1**	**21.2**	**18.5**	**21.7**	**17.6**	**18.9**	**21.9**	**20.4**	**16.7**	**20.0**	**19.5**

Table 2. **SPARF Benchmark on Novel View Synthesis (Normal Test)**. We compare the validation PSNR of some of the widely used novel view synthesis techniques on the SPARF dataset for the generalization of novel view synthesis beyond a single example and on the normal testing-views tracks similar to the ones seen in training views. One view (1V) and three views (3V) inputs are reported, and * indicates retraining the baseline backbone on the high-resolution images of the SPARF dataset.

	1-view			3-view		
Strategy	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
uniform	20.03	0.94	0.10	21.00	0.94	0.09
Q-Gaus.	20.55	0.93	0.09	21.83	0.94	0.08

Table 3. **Effect of Loss Sampling Strategy.**We study the effect of Loss sampling strategy (uniform *vs*. Q-Gaussian) on airplane class.

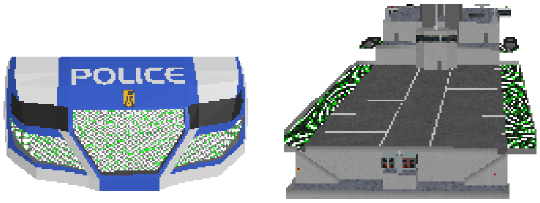

Figure 6. **Rare Cases of Faulty Textures**. Some objects in ShapeNet [2] have doubled textrures in some parts, leading to faulty renderings.

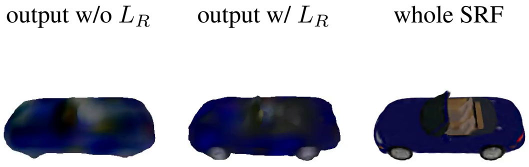

Figure 7. **Effect of the Perceptual Loss** L_R. Adding a perceptual loss on volume-rendered images during training SuRFNet insures the rendered images remain closer to how they should be rendered, as the 3D radiance colors supervision won't guarantee the rendering quality. *(left)*: without perceptual loss , *(middle)*: with the loss.

uniform sampling strategy depletes the learning capacity of the network and can degrade performance. The effect is more evident when the number of views is one, where the partial SRFs are more sparse and the training is delicate. **Loss Hyperparameters.** For the density threshold α_{dense} defined in Eq 2 and 3, the validation accuracies of SuRFNet on car class are 13, 14.7, 67.8, 67, 67, 67.2, 62.5 for the values of -0.01, -0.001, 0, 0.001, 0.003, 0.01, 0.03 of α_{dense} respectively. The hyperparameter σ which governs the spread of the Q-Gaussian loss is studied as follows. the validation accuracies of SuRFNet on airplane class are 52, 70.4, 71.7, 72.1, 72.2, 72.4, 72.3 for the values of 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, and 1.0 of σ respectively. The number of coordinates $\mathbf{c}$ sampled in the Q-Gaussian loss is proportional to the number of coordinates in the input SRFs with multiplier $K = 40$. For different values of this multiplier 1, 5, 10, 20, 40, 80, 200, the validation accuracies of SuRFNet trained on airplane class are 72.2, 72.7, 72.9, 72.5, 71.8, 71.7, and 71.6 respectively.

3.4. Faulty Textures

In some rare instance of the shapes in ShapeNet [2], some objects have doubled textures in some areas. This occurs in less than 1% of the data and leads the renderer to render the background instead in these areas (highlighted with green). See Figure 6 for examples of these cases.

3.5. The Irregularity of SRFs

The optimized whole SRFs used in our training are irregular 3D data structures. They hold many non-empty voxels that contain low-density radiance information that does not affect rendering. As can be seen in Figure 8, the non-empty and low-density components do not affect rendering but contain radiance information (*e.g*. black albedo) that affects the 3D learning pipeline. This motivates the use of the specialized losses proposed in this work, in order to tackle these challenges associated with SRFs.

4. Additional Results

Additional results of normal test tracks benchmark of SPARF are presented in Table 2. Please see figures 12 and 13 for differences between the normal train/test track and the OOD hard track .More comparisons and generations are provided in Figures 17,18,16. Regarding real images, we ahow more Co3D images and their SRFs and PixelNeRF

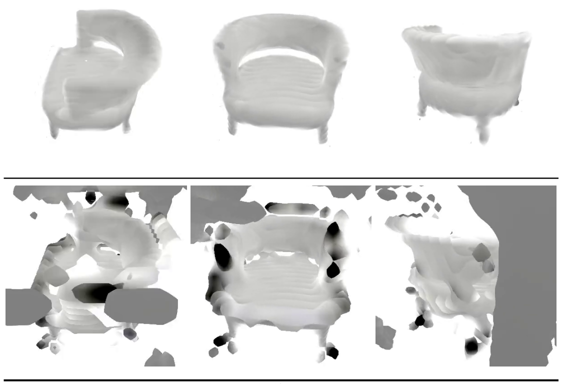

Figure 8. **The Irregularity of SRFs**. The optimized SRFs used in our training are irregular 3D data structures. they include non-empty voxels that contain low-density radiance information that does not affect rendering. *(top)*: renderings of a whole SRF, *(bottom)*: renderings of the same SRF when densifying non-empty voxels.

reconstruction in Figure 19. Note that the goal of this figure is to demonstrate the quality of the renderings form proper SRFs (similar ot the ones used in training SuRFNet), compared to a 2D-based network (like PixelNeRF [14]). It is not used to evaluate generalization ability from few input views on real images, but to show the potential of training on real images.

one important aspect to consider is the *3D consistency* of our SuRFNet renderings compared to the other 2D methods, especially when moving *out-of-distribution of the training views*. This is one of the most important aspects we investigate in our work that previous works in the literature have overlooked. Figure 9 demonstrates that as the testing rendered views move outside of the training distribution (going right), SuRFNet generally produces more consistent renderings than all previous 2D-based methods [[14, 7]].

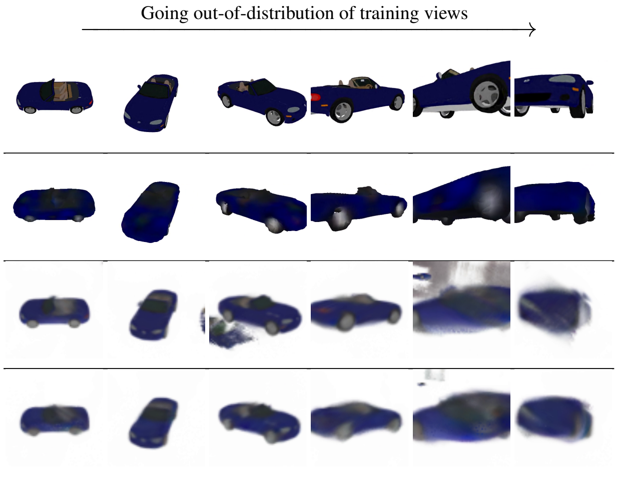

Figure 9. **Going Out-of-distribution of Training Views**. Renderings are shown of ground truth using whole SRFs (*first row*), SurFNet [ours] (*second row*), VisionNeRF [7] (*third row*), and PixelNeRF [1] (*bottom row*). As rendered views move outside of the training distribution (going right), SuRFNet generally produces more 3D-consistent renderings than previous 2D-based methods.

References

[1] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. Pix2nerf: Unsupervised conditional p-gan for single image to neural radiance fields translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3981–3990, June 2022. 5

[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. 4

[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 1

[4] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510, June 2022. 1, 4

[5] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R. Venkatesh Babu. Deligan : Generative adversarial networks for diverse and limited data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[6] Abdullah Hamdi and Bernard Ghanem. Ian: Combining generative adversarial networks for imaginative face generation. *arXiv preprint arXiv:1904.07916*, 2019. 3

[7] Kai-En Lin, Lin Yen-Chen, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *WACV*, 2023. 3, 4, 5, 12, 13

[8] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, aug 1987. 2

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[10] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 6

[11] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10901–10911, October 2021. 14

[12] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-

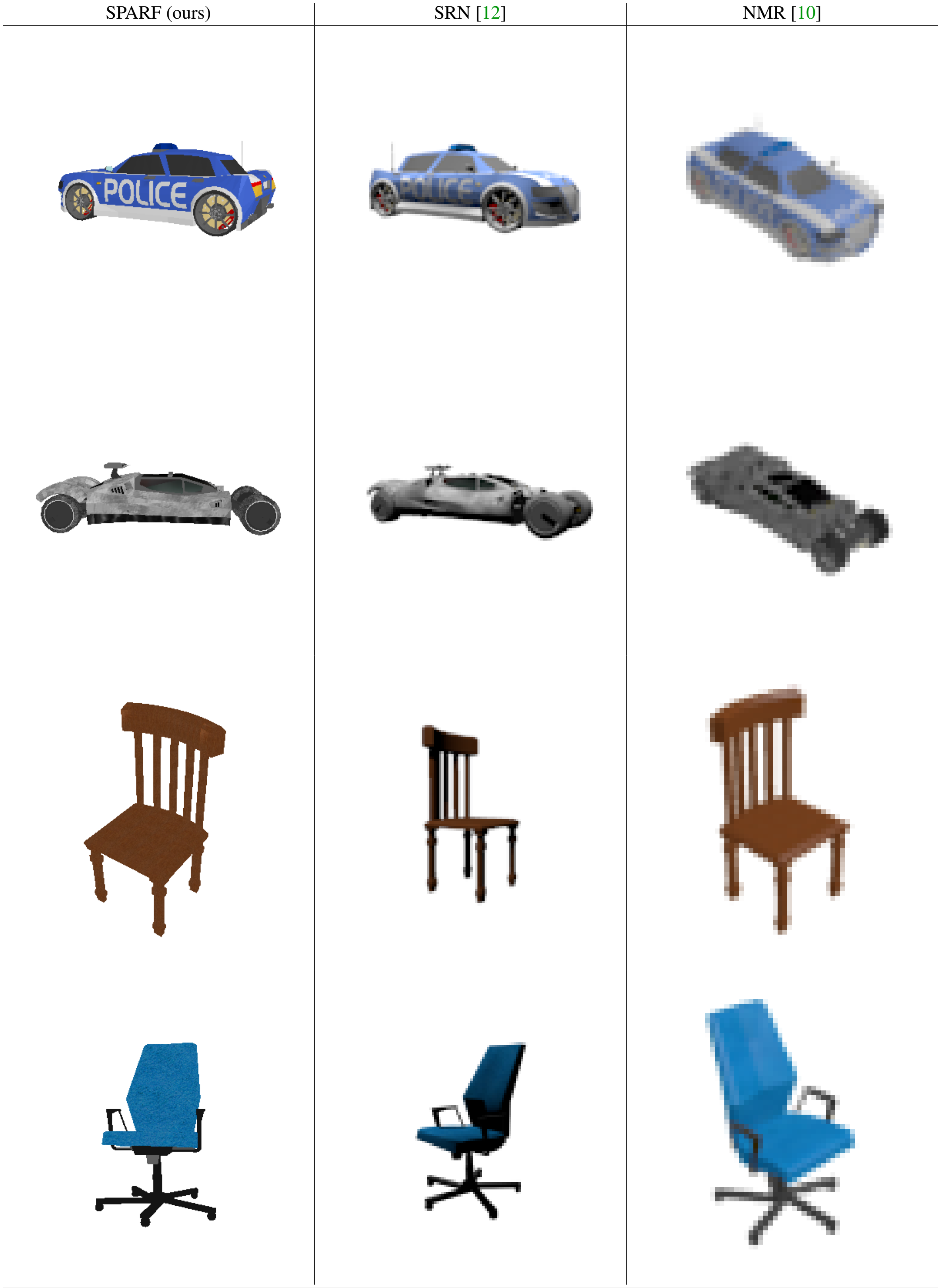

Figure 10. **SPARF *vs*. other Datasets** . SPARF offers a large-scale high-resolution dataset compared to other posed multi-view datasets. Note that SRN [12] has only cars and chairs, while NMR [10] and SPARF has 13 classes.

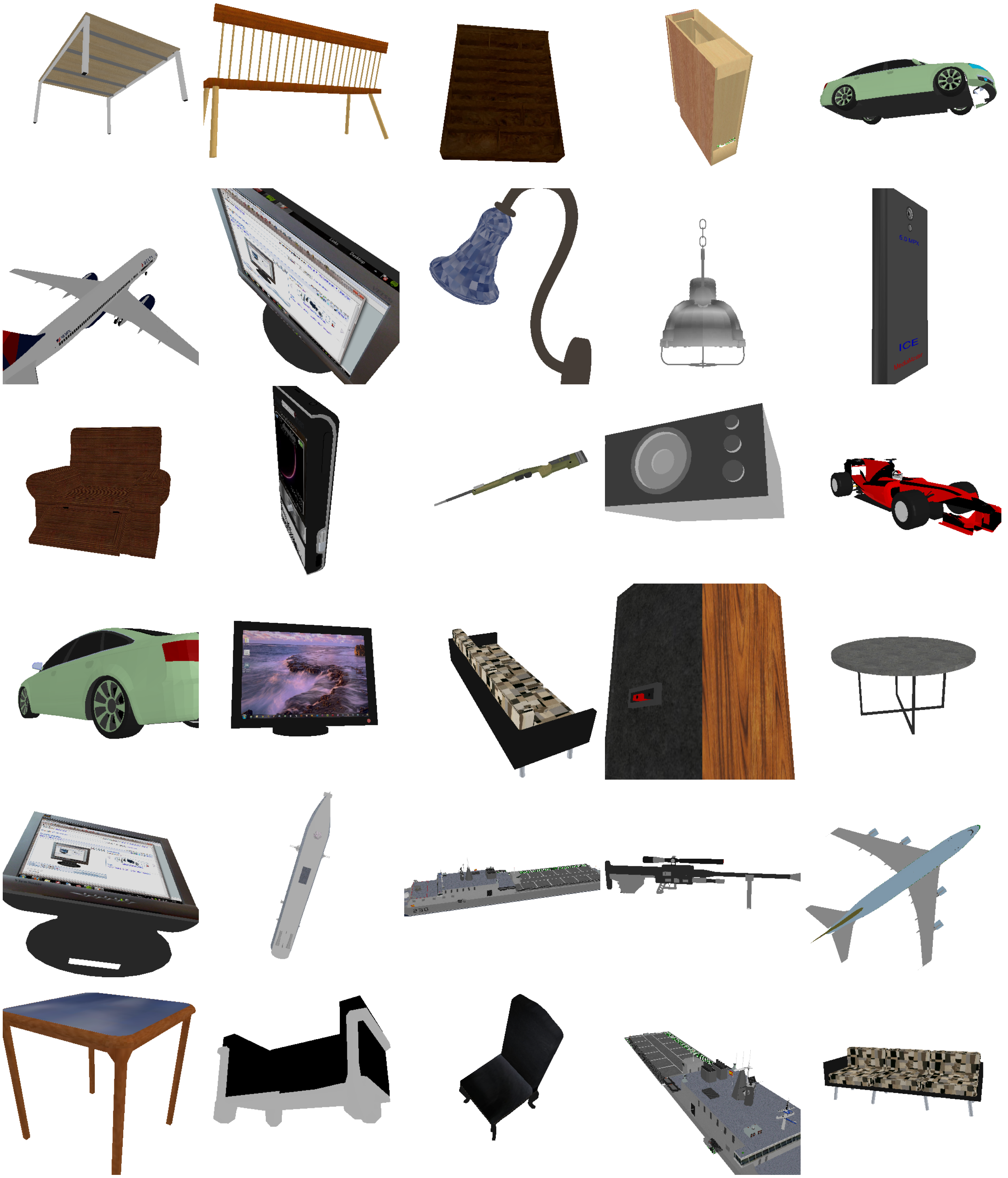

Figure 11. **SPARF: a Large Dataset for 3D Shapes Radiance Fields and Novel Views Synthesis**.

aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[13] Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

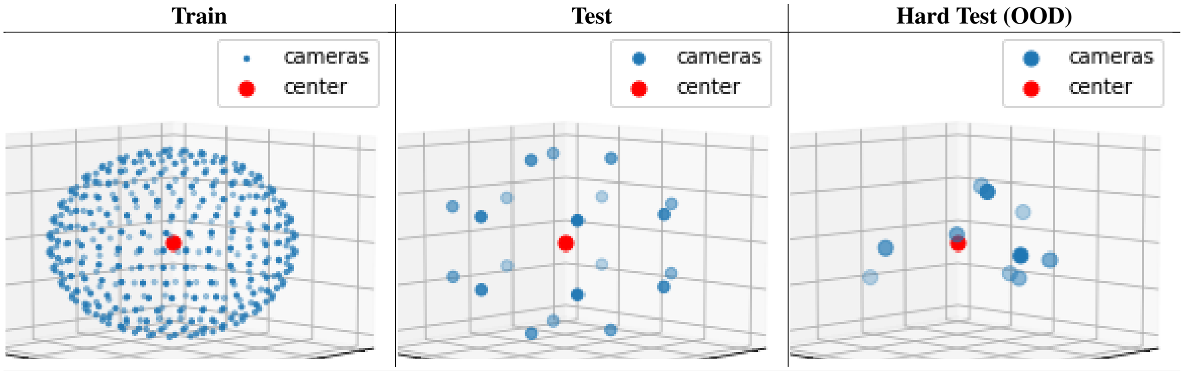

Figure 12. **Cameras Setups for Different SPARF Splits**. Here, we show different visualizations of the camera setups of the three splits of SPARF. (*Train*): 400 determinsitic spherical views, (*Test*): 20 random spherical views, (*hard OOD Test*): 10 random views.

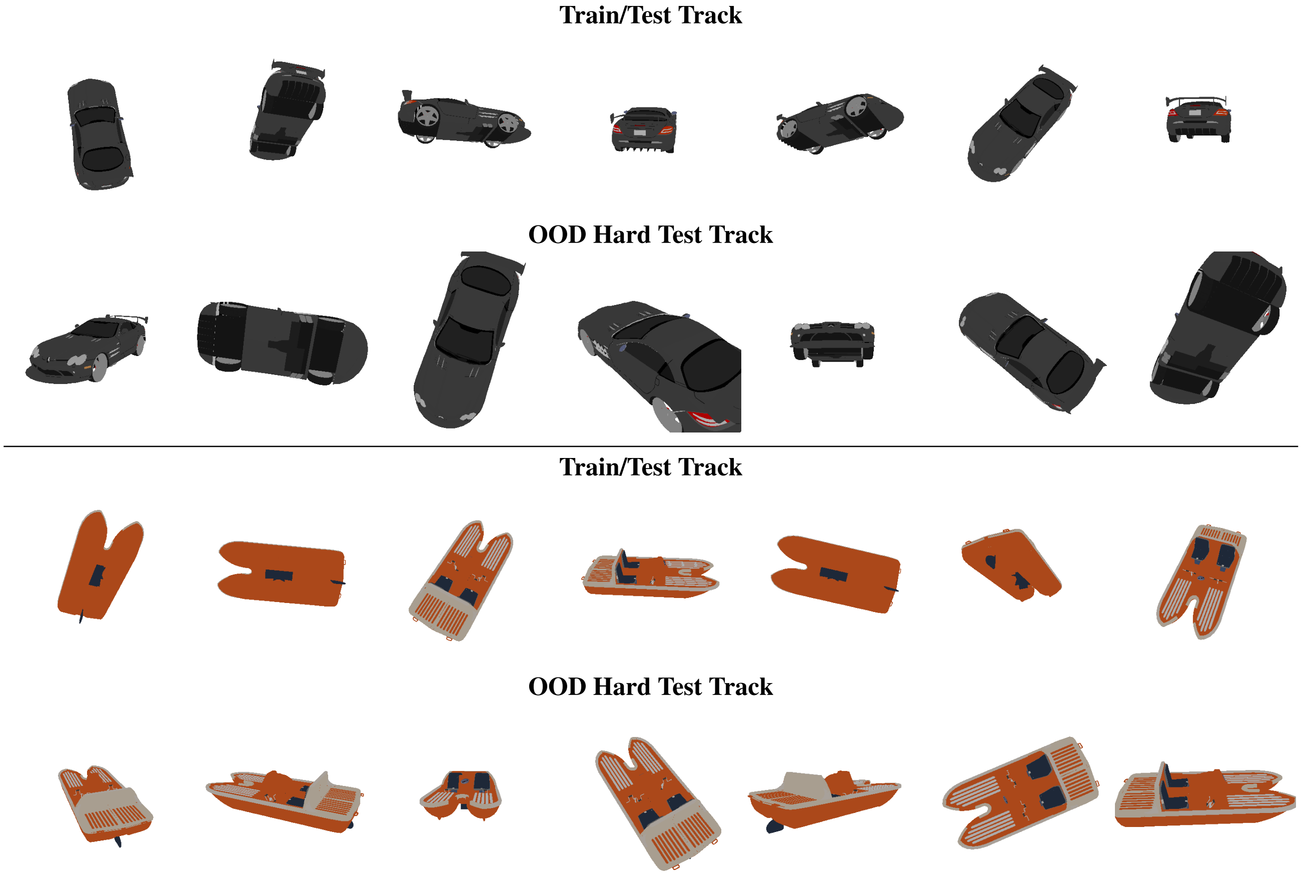

Figure 13. **SPARF Splits**. SPARF has three main splits for every 3D shape: training views (400 views), test views (20 views), and OOD “hard" views (10 views) as can be shown in the examples above.

3

[14] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 3, 4, 5, 12, 13, 14

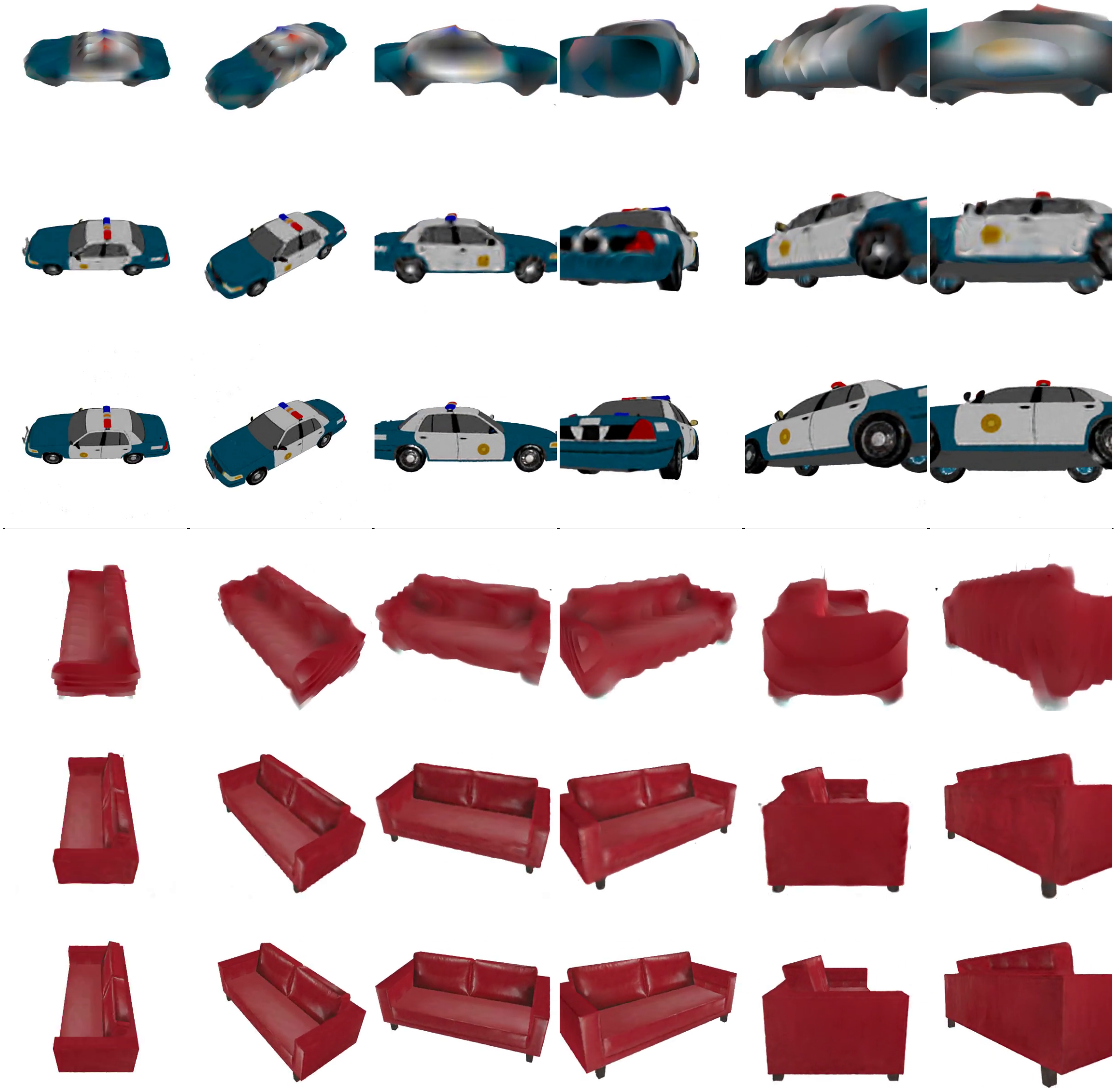

Figure 14. **SRFs: The optimized Sparse Radiance Fields in SPARF 1**. A total of one million SRFs have been collected in SPARF, including on multiple voxel resolutions: 32 (*top*), 128 (*middle*), and 512 (*bottom*) for every 3D shape.

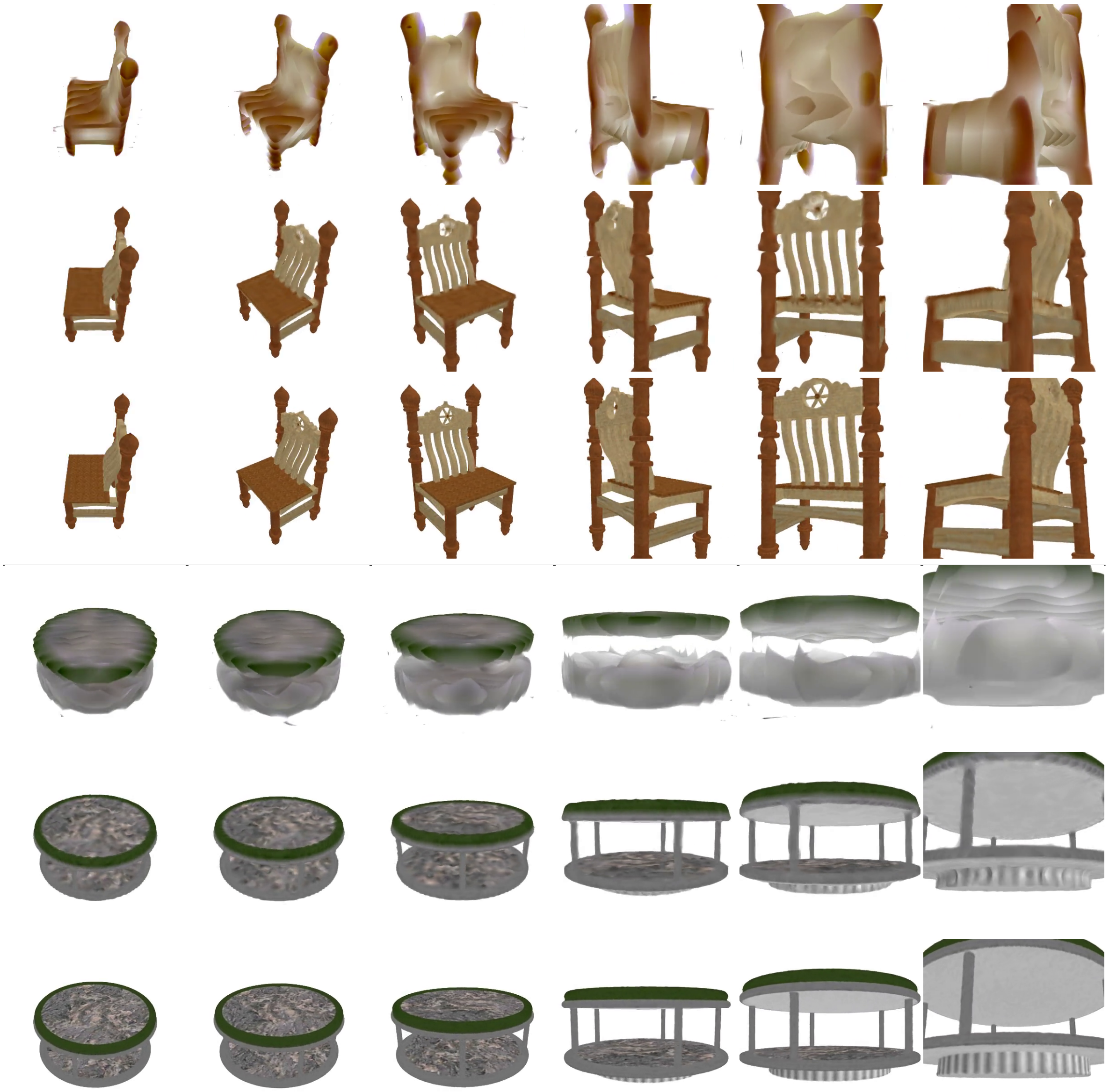

Figure 15. **SRFs: The optimized Sparse Radiance Fields in SPARF 2**. A total of one million SRFs have been collected in SPARF, including on multiple voxel resolutions: 32 (*top*), 128 (*middle*), and 512 (*bottom*) for every 3D shape.

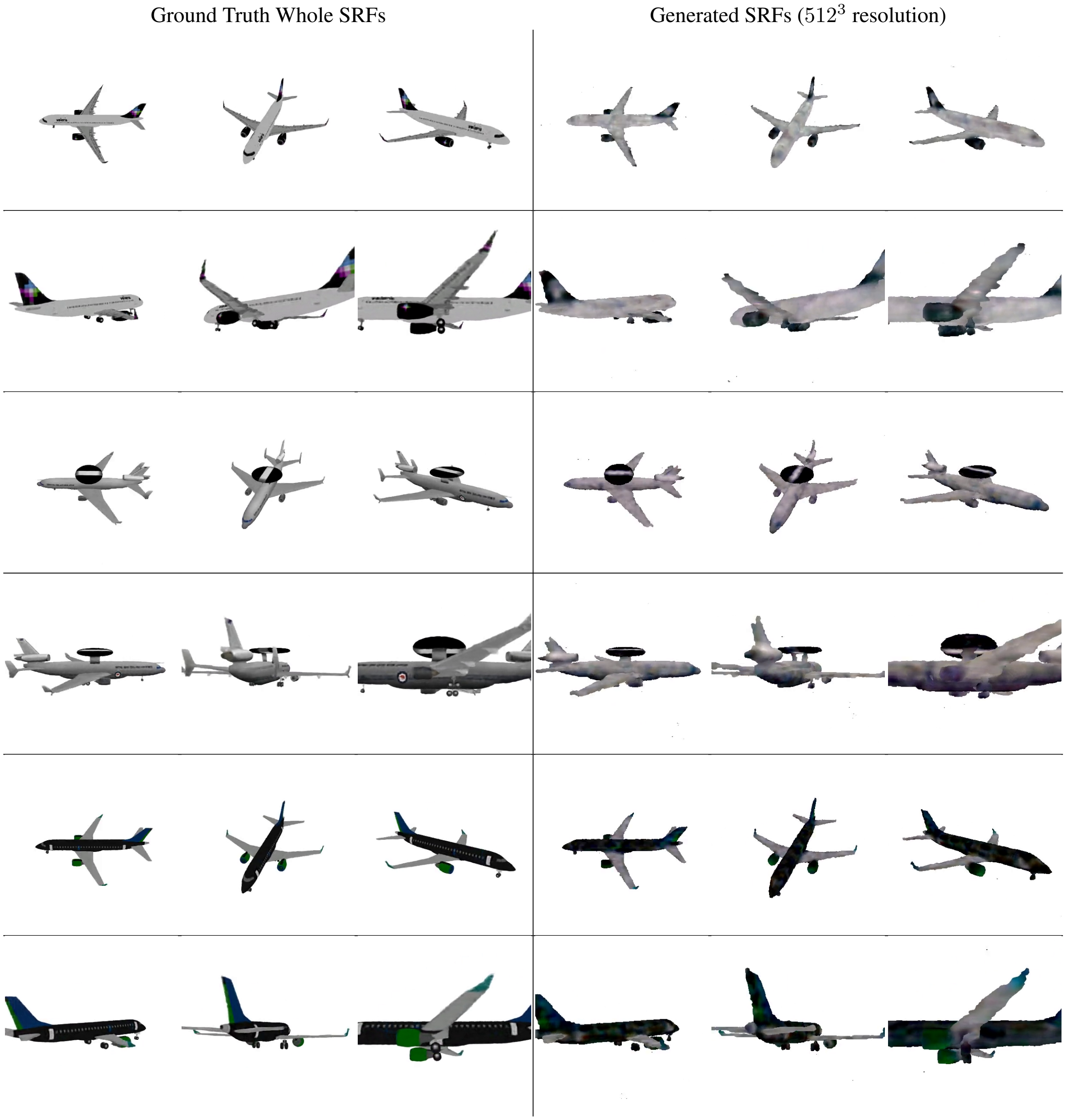

Figure 16. **SuRFNet: Generating High-Resolution Radiance Fields**. We show some volume-rendered sequences based on our SuRFNet voxel radiance field outputs (512 resolution), given only 3 images of each shape. This demonstrates the capability of SuRFNet to generate high-resolution sparse voxel SRFs. Note that, here, SURFNet is overfitting on a small dataset in these examples and is not meant for shape generalization.

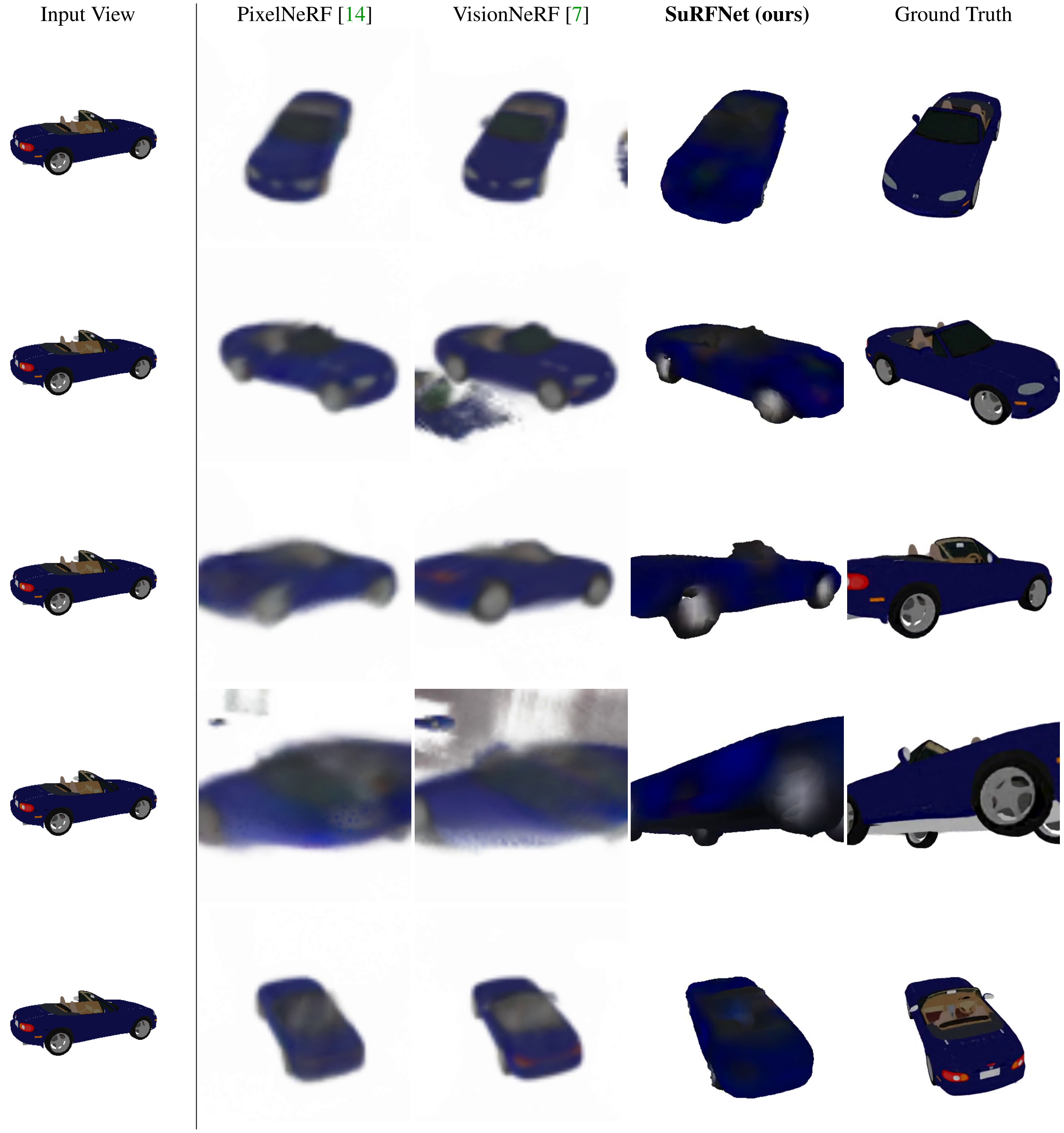

Figure 17. **Qualititve Comparisons 1**. We show different render from our SuRFNet outputs generated from a single image compared to other methods (pixel-Nerf [14], and VisionNerf [7]) and whole SRF ”GT" renderings. Note that the predicted views are outside the training views distribution (zoomed in randomly). This test highlights the weakness of the 2D-based baselines [14, 7] outside the training track, while our 3D approach maintains multi-view consistency everywhere.

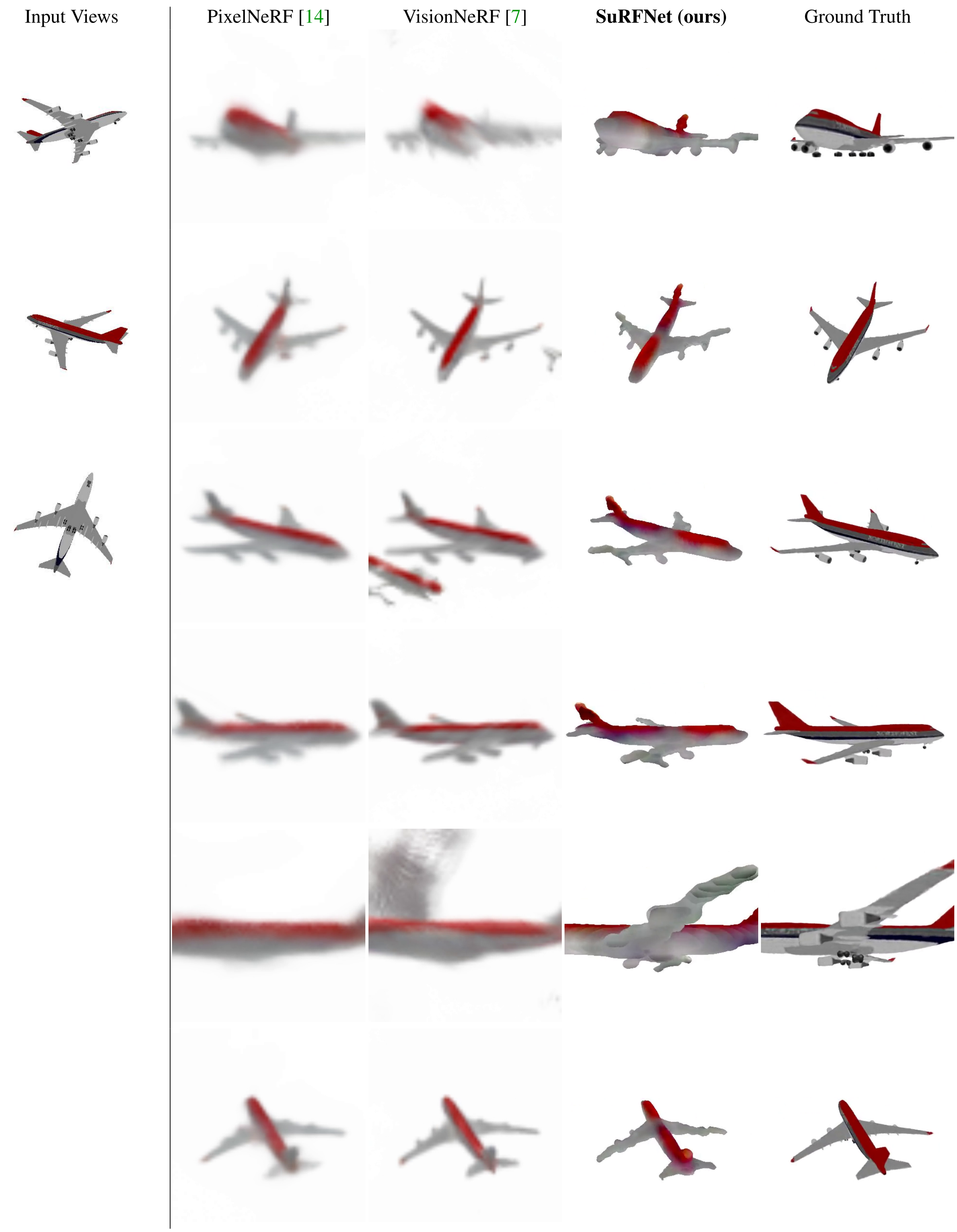

Figure 18. **Qualititve Comparisons 2**. We show different render from our SuRFNet outputs generated from 3 input images compared to other methods (pixel-Nerf [14], and VisionNerf [7]) and whole SRF ”GT" renderings. Note that the predicted views are outside the training views distribution (zoomed in randomly). This test highlights the weakness of the 2D-based baselines [14, 7] outside the training track, while our 3D approach maintains multi-view consistency everywhere.

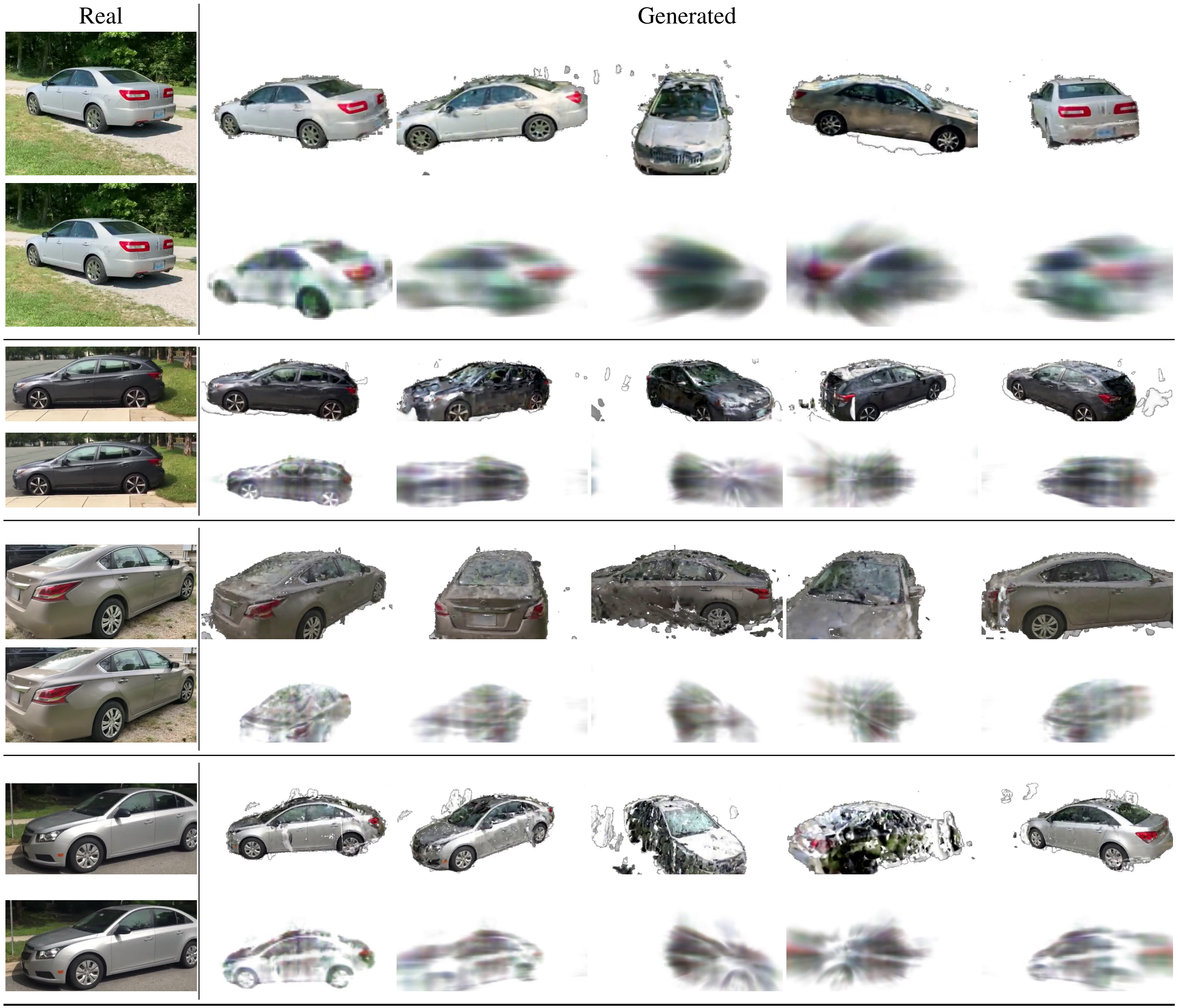

Figure 19. **Real images** We show real images of Co3D [11] (*left*) and the corresponding generated views from our SRFs (*rows' top part*) and pretrained PixelNeRF [14] (*rows' bottom part*).