Supplementary Material for Single-Stage Diffusion NeRF: A Unified Approach to 3D Generation and Reconstruction

Hansheng Chen[1,*] Jiatao Gu[2] Anpei Chen[3] Wei Tian[1] Zhuowen Tu[4] Lingjie Liu[5] Hao Su[4]
[1]Tongji University [2]Apple [3]ETH Zürich
[4]University of California, San Diego [5]University of Pennsylvania

A. Details on Batch-Wise Rendering Loss

During single-stage training and test-time reconstruction, we randomly sample a batch of rays B_{ray} from all available observations for each rendering pass. The actual rendering loss needs to be rescaled to account for the batch size $|B_{\text{ray}}|$.

For single-stage training and test-time *finetuning* based on Adam [6], we rescale the rendering loss to keep its overall magnitude invariant to the batch size $|B_{\text{ray}}|$:

$$\mathcal{L}_{\text{rend}}(\{x_i\},\psi)=\mathop{\mathbb{E}}_{i}\left[\frac{N_{\text{ray}_i}}{|B_{\text{ray}}|}\sum_{j\in B_{\text{ray}}}\frac{1}{2}\left\|y_{ij}^{\text{gt}}-y_\psi\left(x_i,r_{ij}^{\text{gt}}\right)\right\|^2\right], \tag{8}$$

where N_{ray_i} is the total number of observed rays of the i-th scene.

For test-time gradient *guidance*, however, we treat the sampled batch B_{ray} as if it constitutes the full observation set. Thus, the gradients originally defined in Eq. (5) are actually calculated by:

$$g\leftarrow\nabla_{x^{(t)}}\lambda_{\text{rend}}^{B_{\text{ray}}}\sum_{j\in B_{\text{ray}}}\frac{1}{2}\left(\frac{\alpha^{(t)}}{\sigma^{(t)}}\right)^{2\omega}\left\|y_j^{\text{gt}}-y_\psi\left(\hat{x}_\phi(x^{(t)},t),r_j^{\text{gt}}\right)\right\|^2, \tag{9}$$

in which the balanced rendering weight $\lambda_{\text{rend}}^{B_{\text{ray}}}:=c_{\text{rend}}(1-e^{-0.1N_{\text{v}}^{B_{\text{ray}}}})/N_{\text{v}}^{B_{\text{ray}}}$ is determined by the batch-effective number of views $N_{\text{v}}^{B_{\text{ray}}}$ instead of the number of all available views N_{v}, with their relationship defined as:

$$N_{\text{v}}^{B_{\text{ray}}}=\frac{|B_{\text{ray}}|}{N_{\text{ray}}}N_{\text{v}}, \tag{10}$$

where N_{ray} is the total number of observed rays of a test scene.

B. Implementation and Hyperparameters

B.1. Implementation Details

We implement our models using PyTorch and MMGeneration toolkit [1]. Our NeRF renderer is based on a public codebase torch-ngp [13], which employs a density-based grid pruning strategy for efficient real-time rendering.

B.2. Hyperparameters

Table 4 presents the complete list of architecture/training/testing hyperparameters used in our experiments. It is worth noting that we adopt step decay policy for both the learning rate and number of inner loop iterations K_{in} during training.

The major difference between unconditional- and reconstruction-purposed models is the training schedule, where reconstruction-purposed training stops early at 80K iterations, as mentioned in the main paper. Other differences lie in the U-Net dropout rate and latent learning rate, which may have marginal effects on the reconstruction performance.

Regarding the Langevin correction step in the form of $x^{(t)}\leftarrow x^{(t)}-\frac{1}{2}\delta\sigma^{(t)}\hat{\epsilon}+\sqrt{\delta}\sigma^{(t)}\epsilon$ with step size δ and independent noise $\epsilon\sim\mathcal{N}(0,I)$, we observe that this technique is more effective in reconstructing Chairs than Cars. Therefore, to reduce inference time, Langevin correction is not used for SRN Cars dataset. Our intuition is that Chairs dataset exhibits higher variety in geometry, and Langevin correction helps better explore the latent space by injecting random noising during sampling.

B.3. Training and Inference Time

We train all our models using two RTX 3090 GPUs, each processing a batch of 8 scenes. On average, a single outer training step takes around 0.5 sec, 80K iterations take around 11 hours, and 1M iterations cost around 6 days.

Under the unconditional generation setting (50 DDIM steps), sampling a batch of 8 scenes takes 4.63 sec on a single RTX 3090 GPU. Under the reconstruction setting with the same batch size, a single guided DDIM step or Langevin step takes 0.21 sec, and a single outer finetuning step takes 0.28 sec (when $K_{\text{in}}=4$). This sums up to around 23 sec for reconstructing a batch of 8 Cars (single-view), and 102 sec for reconstructing a batch of 8 Chairs (single-view) with ad-

[*]Work done during a remote internship with UCSD.

	Unconditional			Reconstruction		
	Cars (full)	Cars (3-view)	Tables (full)	Cars (full)	Cars (3-view)	Chairs (full)
x shape			3×6×128×128			
Latent dimensionality $\dim(X)$			294912			
U-Net base channels			128			
U-Net channel multiplier			1, 2, 2, 4, 4			
U-Net depth			2			
U-Net attention resolutions			32, 16, 8			
U-Net attention heads			4			
U-Net dropout	0.0	0.0	0.0	0.1	0.1	0.1
Diffusion steps			1000			
Noise schedule			Linear			
Scene batch size $\lvert B_{\text{sc}}\rvert$			16			
Ray batch size $\lvert B_{\text{ray}}\rvert$			4096			
Rendering weight constant c_{rend}			40×2^{-14}			
Diffusion weight constant c_{diff}			4			
SNR power ω	0.5	0.5	0.5	0.5	0.5	0.25
Outer loop iterations K_{out}	1M	2M	1M	80K	80K	80K
Inner loop iterations K_{in}	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 4, & 2\text{K} < k_{\text{out}} \le 100\text{K},\\ 2, & k_{\text{out}} > 500\text{K}.\end{cases}$	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 2, & k_{\text{out}} > 2\text{K}.\end{cases}$	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 4, & 2\text{K} < k_{\text{out}} \le 100\text{K},\\ 2, & k_{\text{out}} > 500\text{K}.\end{cases}$	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 4, & k_{\text{out}} > 2\text{K}.\end{cases}$	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 2, & k_{\text{out}} > 2\text{K}.\end{cases}$	$\begin{cases}16, & k_{\text{out}} \le 2\text{K},\\ 4, & k_{\text{out}} > 2\text{K}.\end{cases}$
Latent base learning rate	0.005	0.005	0.003	0.01	0.01	0.01
Decoder base learning rate	0.001	0.001	0.0006	0.001	0.001	0.001
Diffusion base learning rate	0.0001	0.0001	0.00006	0.0001	0.0001	0.0001
Learning rate multiplier	$\begin{cases}1, & k_{\text{out}} \le 500\text{K},\\ 0.5, & k_{\text{out}} > 500\text{K}.\end{cases}$	$\begin{cases}1, & k_{\text{out}} \le 500\text{K},\\ 0.5, & 500\text{K} < k_{\text{out}} \le 1\text{M},\\ 1, & 1\text{M} < k_{\text{out}} \le 1.5\text{M},\\ 0.5, & k_{\text{out}} > 1.5\text{M}.\end{cases}$	$\begin{cases}1, & k_{\text{out}} \le 500\text{K},\\ 0.5, & k_{\text{out}} > 500\text{K}.\end{cases}$	1	1	1
Ray batch size $\lvert B_{\text{ray}}\rvert$			16384			
DDIM steps	50	50	50	75	75	75
Langevin inner iterations	0	0	0	0	0	5
Langevin step size δ			0.4			
Guidance scale λ_{gd}	-	-	-	3.2×2^{14}	0.8×2^{14}	0.4×2^{14}
Rendering weight constant c_{rend}			40×2^{-14}			
FT Diffusion weight constant c'_{diff}			1			
FT SNR power ω	0.5	0.5	0.5	0.5	0.5	0.25
FT outer loop iterations K_{out}	0	0	0	Table 6	Table 6	Table 6
FT inner loop iterations K_{in}			Table 6			
FT latent base learning rate			Table 6			
FT learning rate multiplier			$0.998^{k_{\text{out}} \cdot K_{\text{in}} + k_{\text{in}}}$			

Table 4. Architecture/training/testing hyperparameters. $k_{\text{out}}, k_{\text{in}}$ correspond to the outer and inner loop iteration indices in Algorithm 1. 2^{14} is the number of pixels per view.

ditional Langevin steps. Once the triplane latent codes are sampled, neural rendering can be performed in real time to synthesize the output images.

C. Additional Model Details

In the interest of reproducibility, this section provides additional details about the models used in our experiments. These techniques were not discussed in the main paper, because they are not essential components of the proposed method, and they seem to have negligible effect on the overall results (Table 5). Nevertheless, we have included them in our implementation to maintain consistency with an earlier version of our codebase where they were found to be useful at one stage.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓
SSDNeRF (standard)	23.52	0.913	0.078	16.39
W/o Tanh	23.59	0.913	0.077	16.34
W/o L2 regularization	23.48	0.913	0.077	16.62

Table 5. Single-view reconstruction results on SRN Cars, showing that Tanh and L2 regularization are likely to be redundant.

C.1. Bounding the Latents via Tanh Mapping

In an earlier version of our implementation of the diffusion model, we use the $\hat{\epsilon}$ prediction format as in DDPM [4] instead of the current $\hat{v}$ format proposed by [10]. To stabilize denoising-based sampling process, the $\hat{\epsilon}$ format requires clipping the denoised prediction $\hat{x}$ at each step, which

N_v	View indices	K_out	K_in	LR	PSNR↑	SSIM↑	LPIPS↓	FID↓
1	64	25	4	0.005	23.52	0.913	0.078	16.39
2	64, 104	50	4	0.01	26.49	0.944	0.054	10.66
4	0, 83, 167, 250	100	4	0.02	28.29	0.955	0.049	11.09
8	0, 36, 71, 107, 143, 179, 214, 250	160	5	0.04	31.26	0.973	0.035	8.54
16	0, 17, 33, 50, 67, 83, 100, 117, 133, 150, 167, 183, 200, 217, 233, 250	200	8	0.08	34.31	0.986	0.018	3.09
32	0, 8, 16, 24, 32, 40, 48, 56, 65, 73, 81, 89, 97, 105, 113, 121, 129, 137, 145, 153, 161, 169, 177, 185, 194, 202, 210, 218, 226, 234, 242, 250	200	8	0.08	35.66	0.989	0.015	2.35

Table 6. Details on sparse-to-dense reconstruction on SRN Cars dataset, including the number of input views N_v and their indices, number of finetuning outer loop iterations K_out, number of finetuning inner loop iterations K_in, finetuning learning rate of the latent code, and novel view synthesis evaluation results.

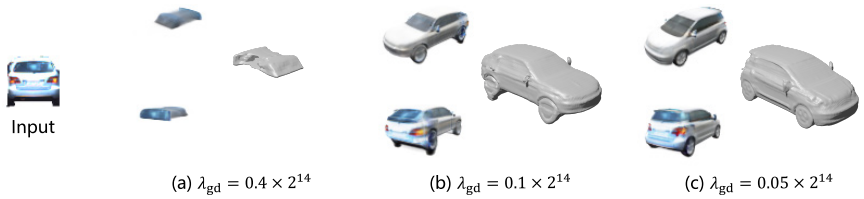

Figure 10. Failure case (a) and (b) in single-view NeRF reconstruction from real images. Sample (c) resolves this issue by reducing the guidance scale λ_gd.

is suitable for bounded data. This motivated us to bound the latent code x_i element-wise via an additional Tanh layer.

Specifically, let $x_i := s \cdot \tanh x_i^\text{raw}$ be the bounded latent code within the interval $(-s, s)$, where x_i^raw denotes a raw, unbounded parameterization of the code. During single-stage training and test-time finetuning, we perform optimization on the leaf variable x_i^raw in the unbounded space. During test-time sampling, the denoised prediction $\hat{x}$ is thus hard-clipped to $[-s, s]$ as well. We set the scale hyperparameter s to 2 in all our experiments.

Because our final models have switched to the $\hat{v}$ prediction format, Tanh mapping may not be an essential component of SSDNeRF, as indicated in Table 5.

C.2. Additional L2 Regularization

L2 latent regularization in auto-decoder training originates from the assumed Gaussian latent prior [9]. In two-stage diffusion NeRF [8] or occupancy field [11] models, L2 regularization helps control the norm of the latent codes and discourage outlying values with respect to the clipping during sampling. During single-stage training and test-time finetuning, we also keep this regularization term in the actual loss function:

$$\mathcal{L} = \lambda_\text{rend}\mathcal{L}_\text{rend}(\{x_i\}, \psi) + \lambda_\text{diff}\mathcal{L}_\text{diff}(\{x_i\}, \phi) + \frac{\lambda_\text{reg}}{\dim(X)} \mathop{\mathbb{E}}_i \left[\|x_i\|_F^2\right], \tag{11}$$

where $\dim(X)$ is the latent dimensionality, and the regularization weight λ_reg is set to 0.003. However, as suggested in Table 5, L2 regularization also has negligible impact under the single-stage training framework.

D. Experiment Details and Additional Results

D.1. Details on Sparse-to-Dense Reconstruction

Table 5 presents more details on the experiment settings, testing hyperparameters, and evaluation results of sparse-to-dense reconstruction on SRN Cars dataset.

Overall, we find that more iterations and higher learning rate are required when finetuning on more input views, but the learning rate should not exceed the upper bound of 0.08 for stability, and a maximum of 200 outer loop iterations (totaling 1600 inner loop iterations) are sufficient for dense-view settings.

D.2. Single-View Reconstruction from Real Images

In this subsection, we provide addition experiments on single-view NeRF reconstruction from real images, using the model trained on the synthetic SRN Cars dataset. This demonstrates the generalization capability of SSDNeRF under substantial domain gap.

Data Preparation We extract images of vehicles from the KITTI 3D object detection dataset [2], which provides annotated 3D bounding boxes of objects in the camera view. We use the provided ground truth bounding box dimensions and poses to align the objects in the same world coordinate system as in SRN Cars dataset. In addition, we leverage the segmentation masks annotated by Heylen *et al.* [3] to remove the background. All images are cropped and resized to 128×128. In real applications, one could also use a monocular 3D object detector and an instance segmentation model to obtain these inputs.

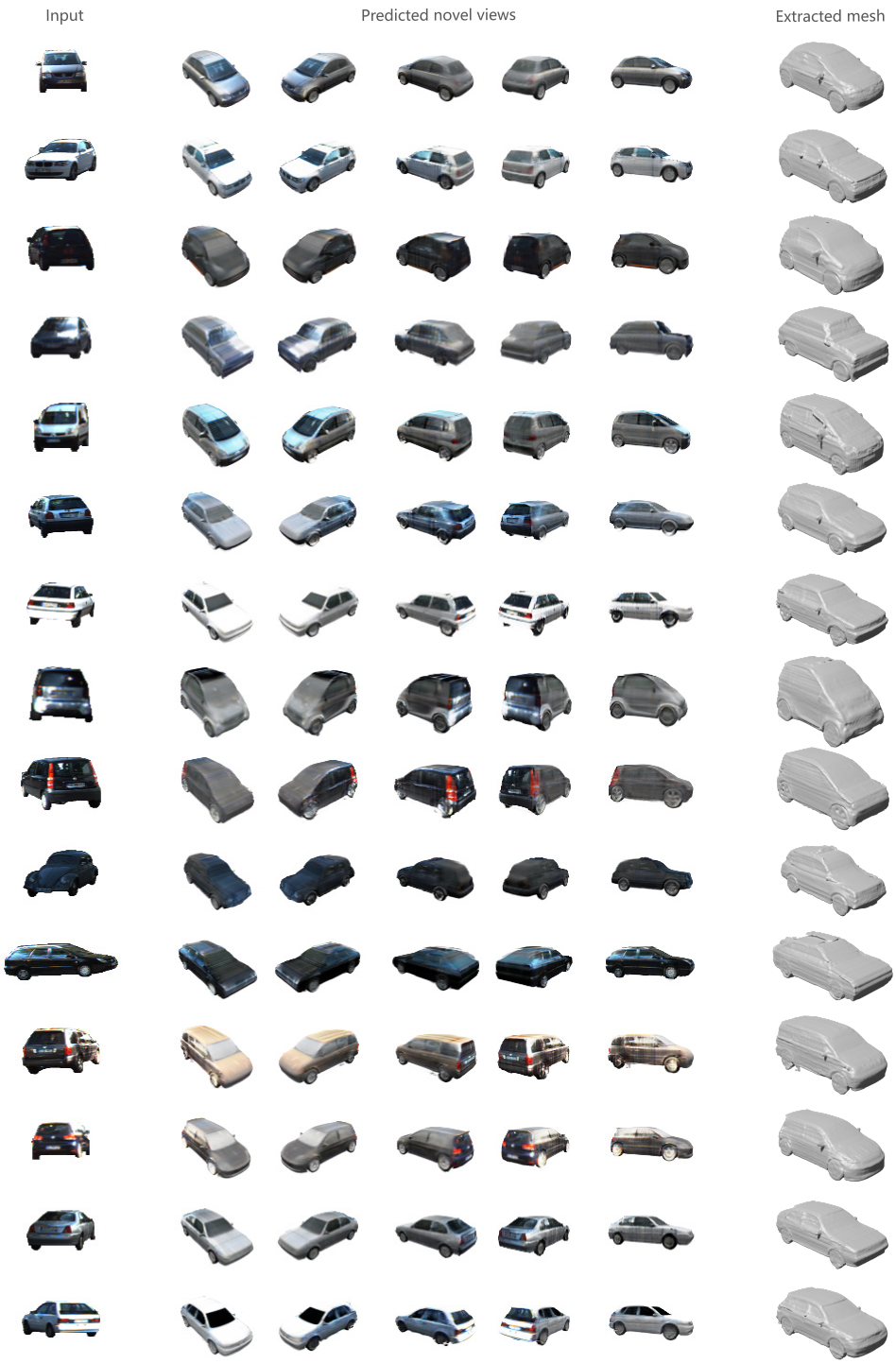

Figure 11. Single-view NeRF reconstruction from real images.

Testing Hyperparameters We enable Langevin correction (5 iterations) to better handle out-of-distribution scenes, and we adopt a different setting of guidance scale $\lambda_{\text{gd}} \coloneqq 0.4 \times 2^{14}$ and finetuning diffusion weight constant $c'_{\text{diff}} \coloneqq 4$.

Qualitative Results and Failure Case We present qualitative examples of novel views and extracted meshes in Figure 11. Apart from that, we have also noticed a failure case where a large portion of the geometry is missing (Figure 10 (a)). Nevertheless, this issue can be resolved by

reducing the guidance scale λ_{gd} (Figure 10 (c)). Overall, we observed that a guidance scale that is too large can result in an unstable sampling process, ultimately leading to corrupted geometries.

D.3. Addition Qualitative Examples

We show randomly sampled scenes generated by SSDNeRF in Figure 12, Figure 13, and Figure 14. For single-view reconstruction, we compare the novel views predicted by SSDNeRF to those predicted by CodeNeRF [5] and VisionNeRF [7] in Figure 15 and Figure 16.

Figure 12. Uncurated samples generated by SSDNeRF trained on SRN Cars dataset.

Figure 13. Uncurated samples generated by SSDNeRF trained on ABO Tables dataset.

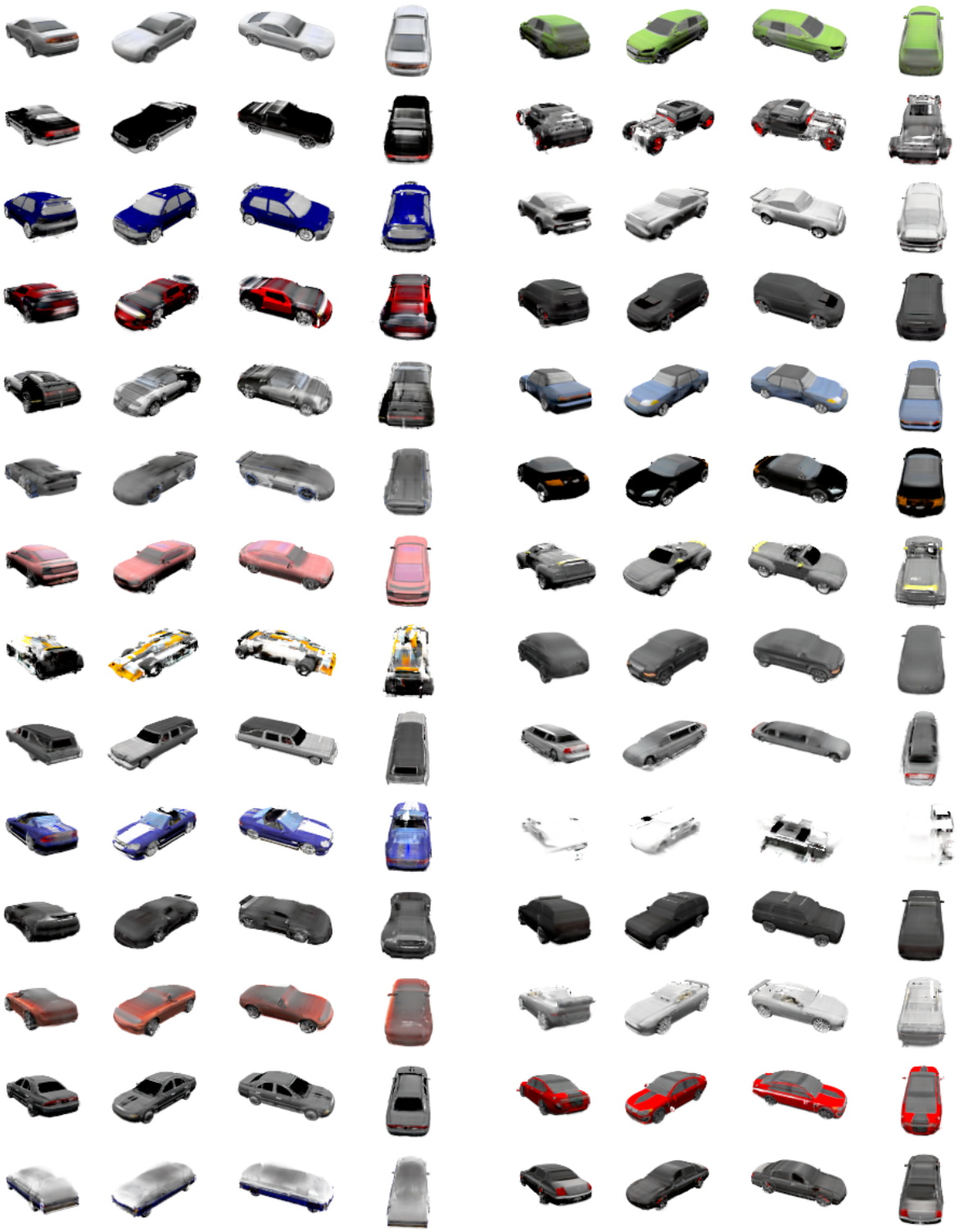

Figure 14. Uncurated samples generated by SSDNeRF trained on a 3-view subset of SRN Cars. Note that the failure case (right column, fifth row from the bottom) is caused by the few outlier training samples, in which the objects are not properly aligned in scale and position due to a data preprocessing issue in SRN Cars [12].

Figure 15. Single-view reconstruction on unseen test objects in SRN Cars.

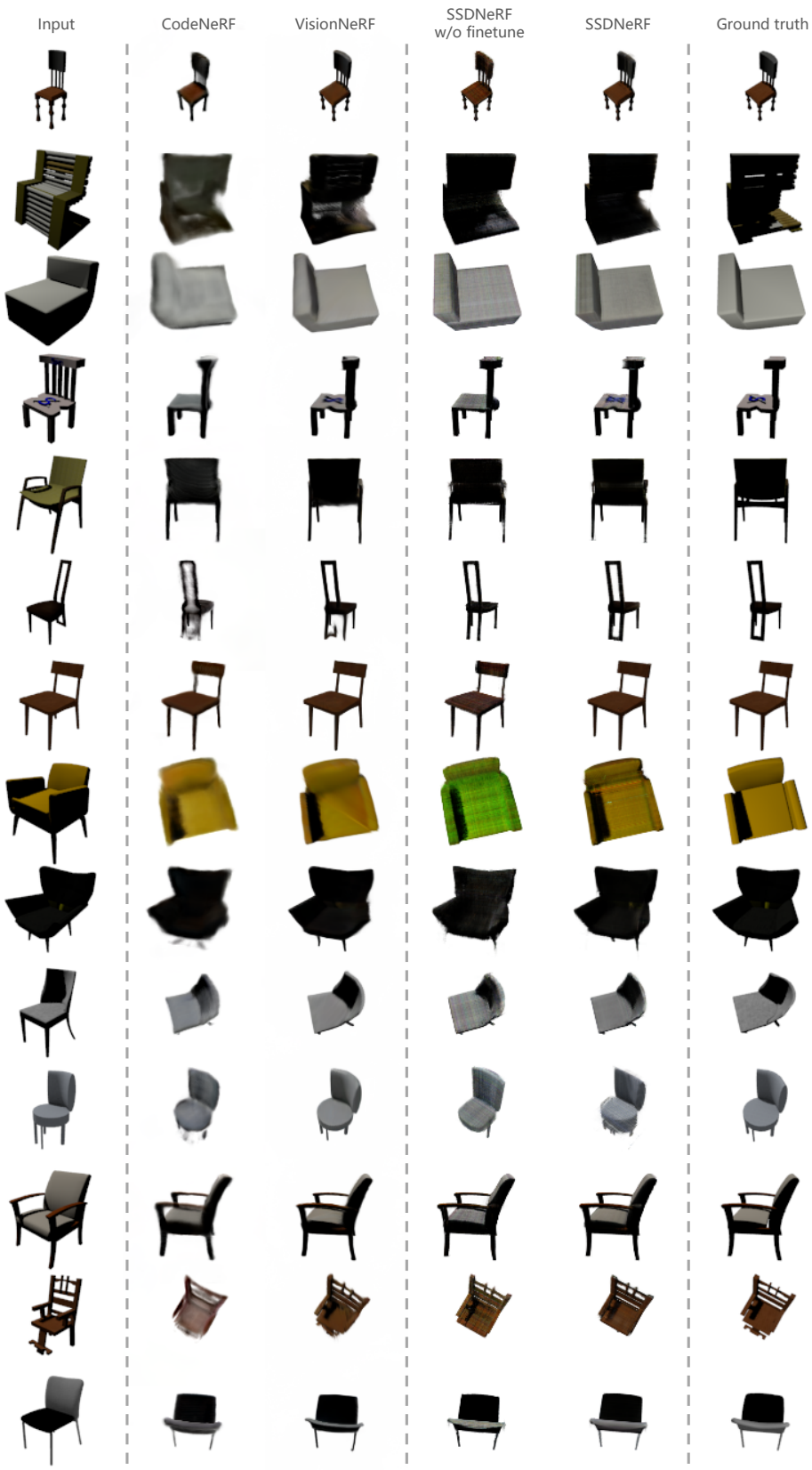

Figure 16. Single-view reconstruction on unseen test objects in SRN Chairs.

References

[1] MMGeneration Contributors. MMGeneration: Openmmlab generative model toolbox and benchmark. https://github.com/open-mmlab/mmgeneration, 2021. 1

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 3

[3] Jonas Heylen, Mark De Wolf, Bruno Dawagne, Marc Proesmans, Luc Van Gool, Wim Abbeloos, Hazem Abdelkawy, and Daniel Olmeda Reino. Monocinis: Camera independent monocular 3d object detection using instance segmentation. In *ICCV Workshops*, 2021. 3

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[5] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *ICCV*, pages 12949–12958, 2021. 5

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[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. 5

[8] Norman Müller, , Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *CVPR*, 2023. 3

[9] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[10] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 2

[11] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *CVPR*, 2023. 3

[12] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 7

[13] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp. https://github.com/ashawkey/torch-ngp, 2022. 1