Mesh2NeRF: Direct Mesh Supervision for Neural Radiance Field Representation and Generation (Supplementary Material)

Yujin Chen[1], Yinyu Nie[1], Benjamin Ummenhofer[2], Reiner Birkl[2], Michael Paulitsch[2], Matthias Müller[2], and Matthias Nießner[1]

[1] Technical University of Munich
[2] Intel Labs

A Implementation Details

A.1 Details of Single Scene Fitting

Datasets. For the ABO dataset, we use twelve objects representing various household categories[3], including a chair (ABO item ID: B075X4N3JH), a table (B072ZMHBKQ), a lamp (B07B4W2GY1), a vase (B07B8NZQ68), a planter (B075HWK9M3), a dumbbell set (B0727Q5F94), a light fixture (B07MBFDQG8), a cabinet (B008RLJR2G), a basket set (B07HSMVFKY), a sofa (B073G6GTK7), and two beds (B075QMHYV8 and B07B4SCB4H). Each object is normalized to [-1,1] in three dimensions. For baselines that require rendered images and camera poses to train, we render each object from cameras distributed on a sphere with a radius of 2.7 using PyTorch3D [1]. We use the lighting defined by PyTorch3D PointLights, with a location in [0, 1, 0], and the ambient component [0.8, 0.8, 0.8], diffuse component [0.3, 0.3, 0.3], and specular component [0.2, 0.2, 0.2]. In our method, rendered RGB images are not used. Instead, we directly utilize the same normalized meshes and the light information.

For the Poly Haven dataset, we use six realistic models[4], including a potted plant, a barber chair, a coffee chart, a chess set, a concrete cat statue, and a garden hose. Rendered images for baseline NeRF training inputs are normalized to the [-1,1] cube. We render each object from cameras distributed on a sphere with a radius of 2.0. The point light is located at [0, 2, 0], with ambient, diffuse, and specular components set to [1.0, 1.0, 1.0], [0.3, 0.3, 0.3], and [0.2, 0.2, 0.2], respectively. Mesh2NeRF training employs the same normalized meshes and lighting. For the SketchFab scene, we use Entrée du château des Bois Francs and Château des Bois Francs . The rendering settings remain consistent with Poly Haven data, with a camera distance of 1.5.

Training. We implement our method in Python using the PyTorch framework. We leverage PyTorch3D [1] and Open3D [5] for processing mesh data and performing essential computations in Mesh2NeRF. Training on both the ABO and

[3] https://amazon-berkeley-objects.s3.amazonaws.com/index.html
[4] https://polyhaven.com/models

Poly Haven datasets comprises 50,000 iterations with a batch size of 1024 rays. We employ the Adam optimizer [2] with a 1cycle learning rate policy [4], where the max learning rate is set to 1×10^{-3}, the percentage of the cycle by 0.001, and the total number of steps is 50,000. For each sampled point, we assign a weight w_{color} of 1 if the sample lies within the surface thickness; otherwise, w_{color} is set to 0. Additionally, we fix $w_{integral}$ at 10 during the training process.

Sampling. During training, we employ ray casting to compute the intersection distance for each ray and the input mesh. For each ray, we conduct stratified sampling, obtaining 512 points distributed along the ray. In the case of rays intersecting the surface, an additional 512 points are sampled within the surface thickness using a random sampling approach. For rays that do not intersect the surface, we randomly sample another set of 512 points along the ray. For each sampled point, Mesh2NeRF can extract ground truth density (or alpha value) and color information directly from the input mesh. This procedure ensures the availability of accurate supervision data for training. In the inference phase, given a specific view, we utilize the camera's intrinsic parameters to determine the origin and direction of the ray. Subsequently, we uniformly sample points along the ray within the object cube. In a scenario with a surface thickness of 0.005, we sample 800 points along each target ray.

A.2 Details of Conditional Generation

Data setting. In the main manuscript, we evaluate NeRF conditional generation on ShapeNet Cars and Chairs, and additionally perform inference on KITTI Cars. For ShapeNet Cars and Chairs, the model is trained ontheir respective training sets. During inference, for the 1-view setup, we use view 64 as the NeRF input and evaluate the other 249 views out of 250. For the 2-view setup, we use views 64 and 104 as the NeRF input and evaluate the other 248 views. We additionally evaluate the results of 3-view and 4-view setups in Section B.2. For the 3-view setup, we use views 0, 125, and 250 as the NeRF input and evaluate the other 247 views. For the 4-view setup, we use views 0, 83, 167, and 250 as the NeRF input and evaluate the other 246 views.

Training and evaluation. For SSDNeRF results, we adhere closely to its official implementation[5] during model training, ensuring consistency with our training data. The evaluation employs the same test set as SRN [3]. For Mesh2NeRF, Blender is utilized to obtain color and intersection distance information for each viewpoint. Additional details are available in the ShapeNet rendering repository[6]. In Mesh2NeRF supervision during training, we use a surface thickness of 0.01. Along each ray, we employ a stratified sampling strategy to distribute 32 points along the ray, along with an additional random sampling of 32 points within the surface thickness. Both w_{color} and $w_{integral}$ are set to 1. In the inference stage, we evenly sample 400 points along each target ray for volume rendering.

[5] https://github.com/Lakonik/SSDNeRF

[6] https://github.com/yinyunie/depth_renderer

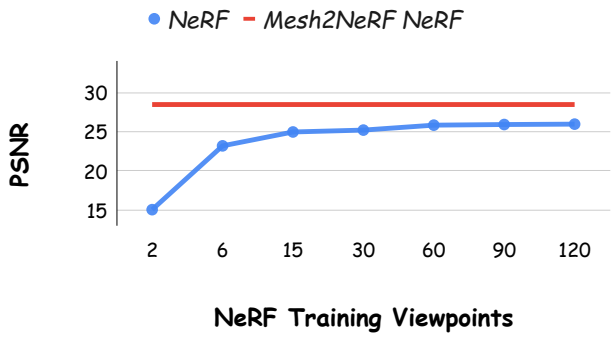

Fig. 9: Comparing Mesh2NeRF with NeRF trained with the different viewpoint numbers. The performance of NeRF saturates as the number of viewpoints increases, *i.e.*, adding viewpoints results in diminishing returns at some point. Overall, the performance is worse than Mesh2NeRF, which does not require renderings and is aware of the whole scene content.

Table 3: Ablation of integral loss $L_{integral}$ of Mesh2NeRF NGP on the Poly Haven dataset.

	Variant	Potted Plant	Barber Chair	Coffee Chart	Chess Set	Cat Statue	Garden Hose	*Average*
PSNR ↑	V1	11.53	27.42	11.90	7.02	12.42	7.09	12.90
	V2	19.26	23.43	21.74	22.62	23.91	20.59	21.93
	V3	**23.96**	25.76	24.93	25.32	24.53	22.68	24.53
	V4	23.66	**26.79**	**26.96**	**26.00**	**24.64**	**23.77**	**25.30**
	V5	13.67	7.08	11.91	25.95	14.84	8.73	13.70
SSIM ↑	V1	0.657	**0.926**	0.739	0.563	0.485	0.423	0.632
	V2	0.744	0.892	0.849	0.851	0.676	0.731	0.791
	V3	0.819	0.900	0.881	0.873	**0.689**	0.750	0.819
	V4	**0.829**	0.907	**0.901**	**0.877**	0.683	**0.756**	**0.825**
	V5	0.642	0.535	0.739	0.884	0.556	0.471	0.638
LPIPS ↓	V1	0.448	0.096	0.446	0.596	0.618	0.620	0.477
	V2	0.123	**0.078**	0.110	**0.076**	**0.236**	**0.106**	**0.122**
	V3	0.088	0.100	0.108	**0.076**	0.261	0.114	0.125
	V4	**0.087**	0.101	**0.105**	0.084	0.268	0.128	0.129
	V5	0.353	0.548	0.446	0.087	0.459	0.514	0.401

Traning and inference time. We train our generative model using two RTX A6000 GPUs, each processing a batch of 8 scenes. On average, 80K training iterations for ShapeNet experiments take around 40 hours, and 10K training iterations for Objaverse mugs take around 5 hours. For inference, under the unconditional generation setting using 50 DDIM steps, sampling a batch of 8 scenes takes 4 sec on a single RTX A6000 GPU. And under the conditional generation setting, and 130 sec for reconstructing a batch of 8 scenes from a single-view condition.

Table 4: Mesh2NeRF NeRF results vary with defined surface thickness on the ABO dataset. Smaller thickness yields better MLP optimization performance, with 0.0050 and 0.0025 providing comparable results. We choose 0.0050 as the default thickness for subsequent experiments.

Surface Thickness	PSNR ↑	SSIM ↑	LPIPS ↓
0.0200	27.18	0.919	0.058
0.0100	27.18	0.929	0.053
0.0050	28.40	0.933	**0.049**
0.0025	**28.93**	**0.934**	0.056

B Additional Results

B.1 More Results on Single Scene Fitting

Impact of integral loss. In this ablation study, we investigate the impact of the integral loss during optimizing neural radiance fields in Mesh2NeRF. We compare five variants of our method, denoted as (V1) through (V5), except for (V1), we maintain $w_{alpha} = 1$ and $w_{color} = 1$. (V1): only uses the integral loss $\mathcal{L}_{integral}$, without the alpha loss $\mathcal{L}_{alpha}$ and the color loss the integral loss $\mathcal{L}_{color}$; (V2): excludes the integral loss $\mathcal{L}_{integral}$; (V3) with the integral loss and uses a weight of $w_{integral} = 1$; (V4) with the integral loss and uses a weight of $w_{integral} = 10$. (V5) with the integral loss and uses a weight of $w_{integral} = 100$. We evaluate the PSNR, SSIM, and LPIPS of each sample test view and provide overall average results across all test views in the dataset. As shown in Table 3, (V4) achieves the highest average PSNR and SSIM, while (V2) exhibits slightly superior performance in terms of LPIPS. Consequently, we select V4 as the default configuration for our Mesh2NeRF.

Comparing using NeRFs with different numbers of viewpoints. We compare Mesh2NeRF NeRF with NeRF trained using varying numbers of views on the chair object from the ABO dataset. In Fig. 9, we chart the mean PSNR for NeRF w.r.t. training views rendered from the mesh, and Mesh2NeRF, directly optimized from mesh. The evaluation results of NeRF exhibit improvement from a few viewpoints to dozens of views (*e.g.*, covering most of the surface), converging with increasing views. Mesh2NeRF captures more comprehensive object information even compared to NeRF at convergence, acting as an upper bound.

Ablation study on surface thickness. In Table 4, we self-compare surface thickness as a hyperparameter defining mesh resolution. For objects normalized to [-1,1] in three dimensions, we vary the surface thickness from 0.02 to 0.0025. Results show average PSNR, SSIM, and LPIPS for test views at 256×256 resolution. A smaller thickness captures finer details but demands denser sampling for rendering. Consequently, we set 0.005 as the default surface thickness for our method.

Lighting model. We apply Phong model in our experiments, while Mesh2NeRF can accommodate any BRDF and lighting. The only adjustment required is to

Table 5: Comparison with Instant NGP with additional depth supervision. Ours (Mesh2NeRF NGP) outperforms Instant NGP in both limited and dense view settings.

Method	Depth loss	PSNR
Instant NGP (limited views)	No	15.75
	Yes	16.19 *(+0.44)*
Instant NGP (dense views)	No	19.51
	Yes	19.72 *(+0.21)*
Mesh2NeRF NGP	-	20.42

change the function computing the color value of the ray hit point. For instance, if Spherical Harmonics lighting replaces the point lighting, the analytic solution (as illustrated in Fig. 3) can also achieve an SSIM over 0.99.

Baseline with geometry information. We compare NeRF baseline with and without GT depth supervision during fitting on the Sketchfab scene Entrée du château des Bois Francs. Results (Table 5) show improvements in Instant NGP with depth supervision in both dense (same as Sec. 4.1) and sparse (30 views) settings. More improvement from depth is gained in the limited views setup (+0.44 PSNR vs. +0.21). Even with depth, Instant NGP is still worse than Mesh2NeRF NGP.

More qualitative comparsions. We present qualitative results from the Sketchfab dataset in Fig.10. Renderings from two views for each object are compared with NeRF baselines, accompanied by corresponding PSNR values. Similarly, Fig.11 showcases qualitative results on the ABO dataset, with renderings from two views for each object compared alongside corresponding LPIPS values. These figures demonstrate that Mesh2NeRF NeRF outperforms NeRF baselines.

High-resolution volume rendering. Our method is not constrained by resolution during rendering, as evidenced by our evaluation setting. In Fig. 12, we compare Mesh2NeRF NGP and Instant NGP renderings at resolutions of 1024×1024. Our approach consistently outperforms Instant NGP, demonstrating its ability to achieve high-resolution results during inference.

B.2 More Results on Conditional Generation

More spare-view NeRF conditional generation results. We present qualitative comparisons of NeRF generation conditioned on sparse-view images on previously unseen objects in ShapeNet Cars and Chairs between SSDNeRF and our method as the supervision. In Figure 13, we showcase the results of NeRF generation conditioned on 3-view inputs in both ShapeNet Cars and Chairs. For a broader perspective, Figure 14 illustrates the results of NeRF reconstructions from 4-view inputs in both ShapeNet Cars and Chairs. In Figure 15, we provide addition NeRF generation results conditioned on single-view KITTI Cars real images, utilizing the model trained on the synthetic ShapeNet Cars. This

highlights the generalization capability of Mesh2NeRF supervision in NeRF generation tasks, even when faced with significant domain gaps.

Study on early-stage results. Our method offers direct supervision to the 3D radiance field, facilitating faster model convergence during training. To illustrate this, we compare the single-view reconstruction results for chair samples at 10,000 iterations (out of a total of 80,000 iterations). As shown in Figure 16, our reconstructed novel views demonstrate improved accuracy and reduced floating noise.

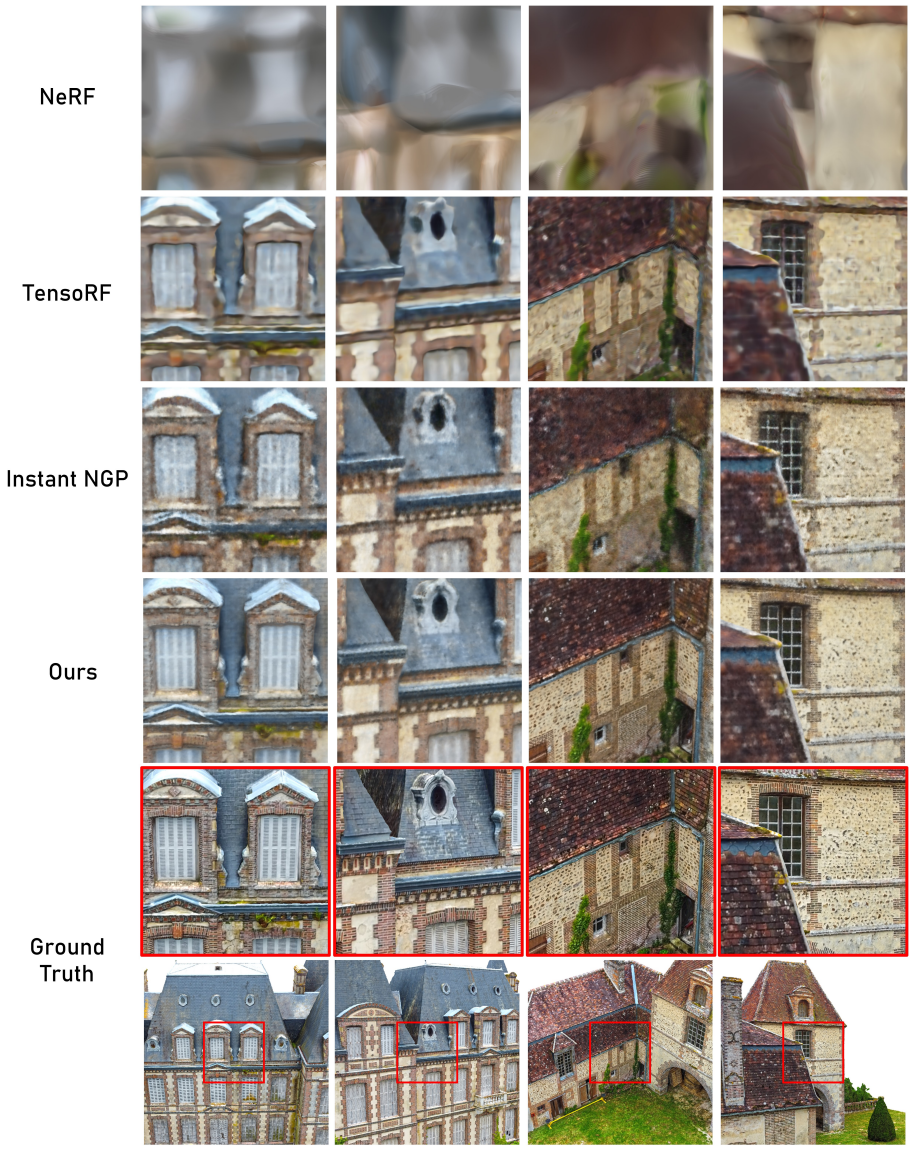

Fig. 10: Comparison on test views for scenes from the Sketchfab dataset. Ours (Mesh2NeRF NGP) outperforms NeRF baseliens in the displayed challenging scenes.

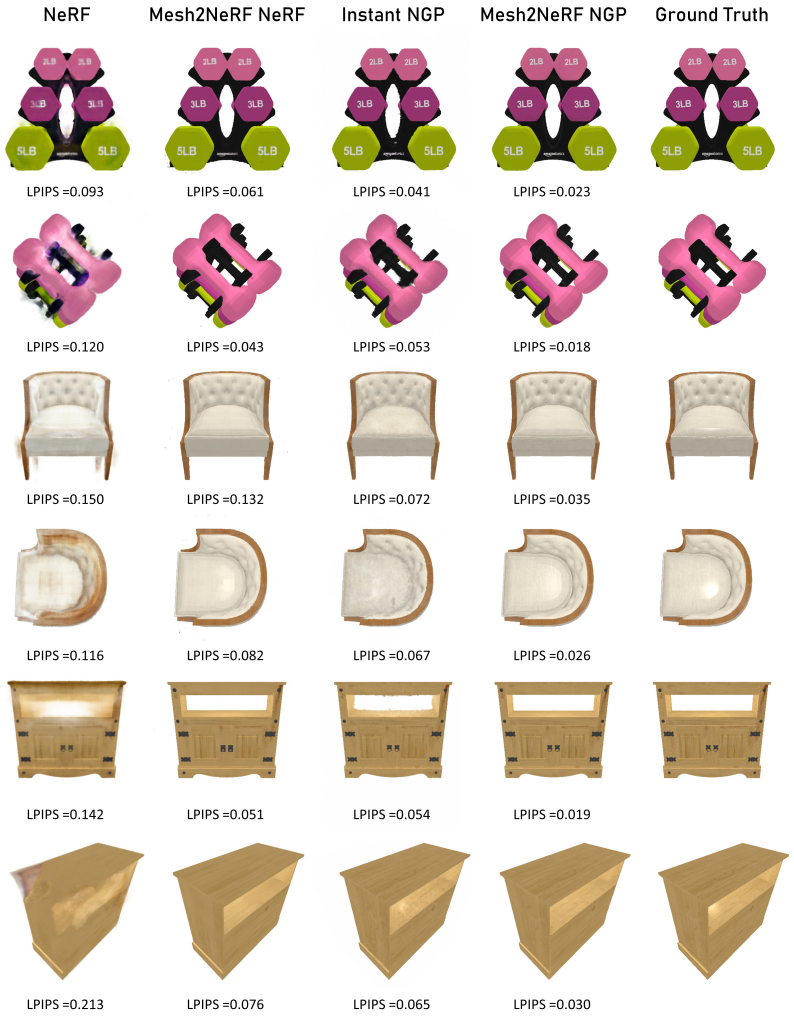

Fig. 11: Comparison on test views for scenes from ABO dataset. For every visualized object, we show two renderings from each method. Our results (Mesh2NeRF NeRF vs. NeRF and Mesh2NeRF NGP vs. Instant NGP) are more accurate and capture finer details in renderings compared to the baselines using the same network architecture.

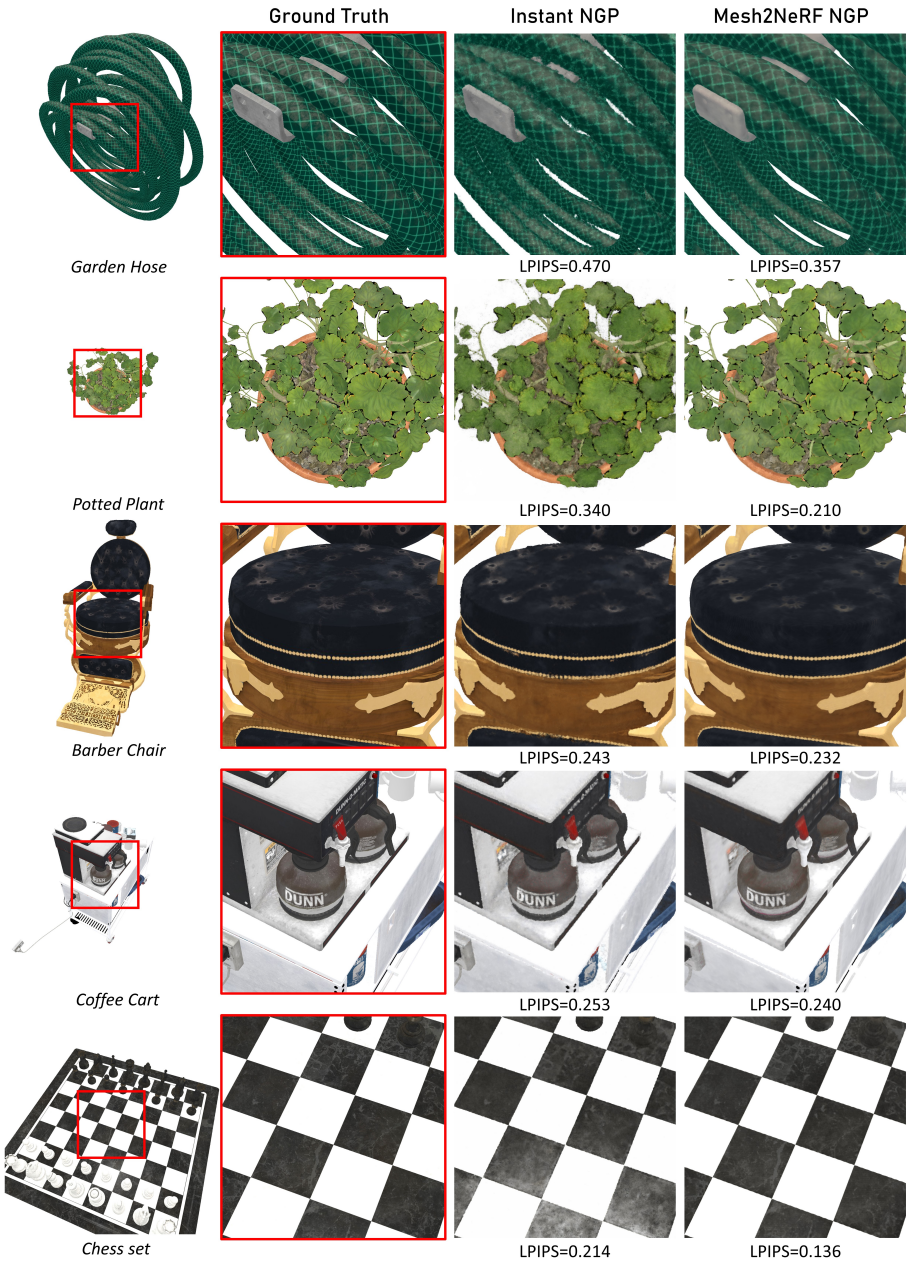

Fig. 12: Comparison on test views for scenes from the Poly Haven dataset at a resolution of 1024 × 1024. Without modifying the training process, Mesh2NeRF NGP outperforms Instant NGP in generating high-resolution renderings.

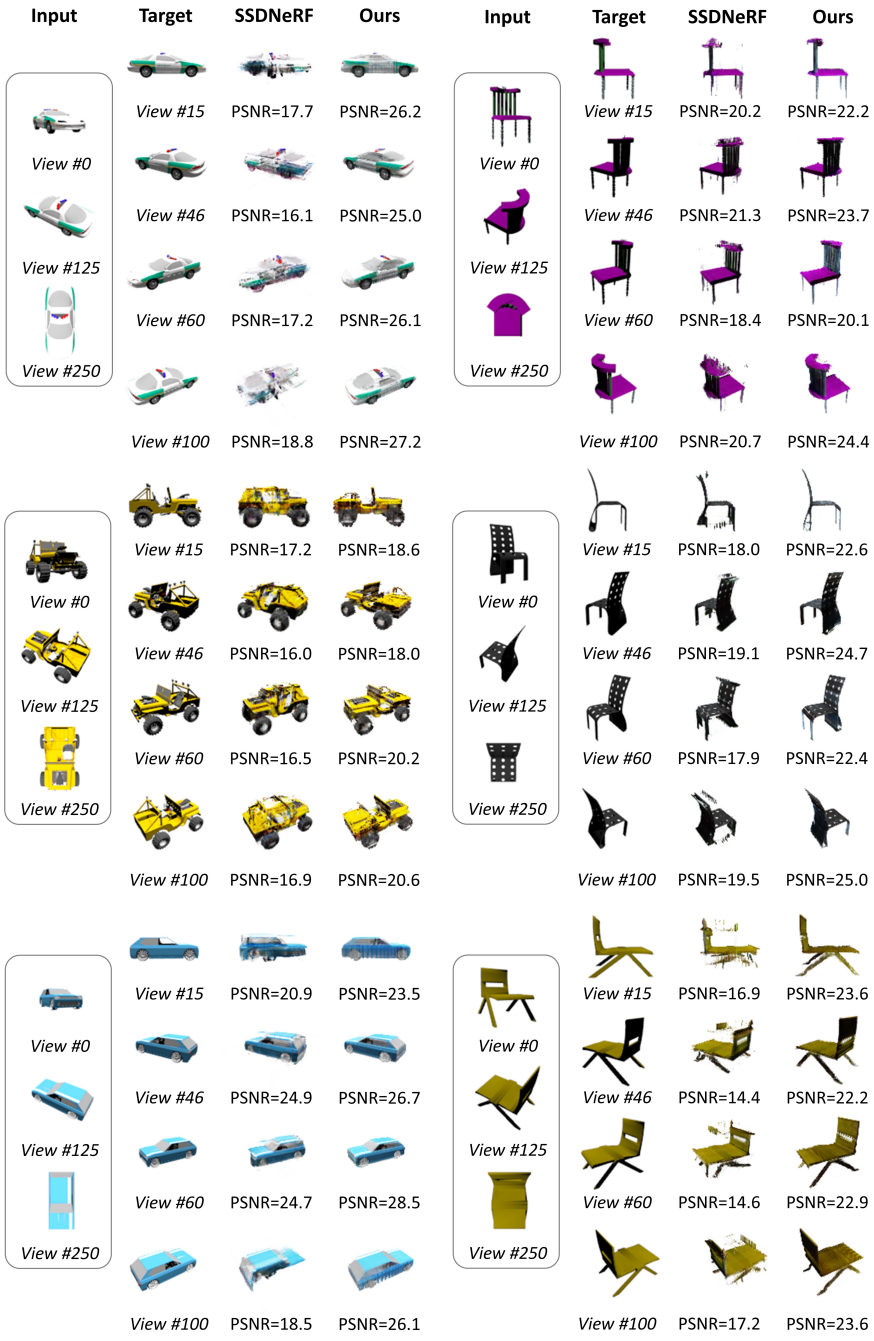

Fig. 13: Qualitatively comparison of NeRF generation conditioned on 3-view input on unseen objects in ShapeNet Cars (left part of the figure) and Chairs (right part).

Fig. 14: Qualitatively comparison of NeRF generation conditioned on 4-view input on unseen objects in ShapeNet Cars (left part of the figure) and Chairs (right part).

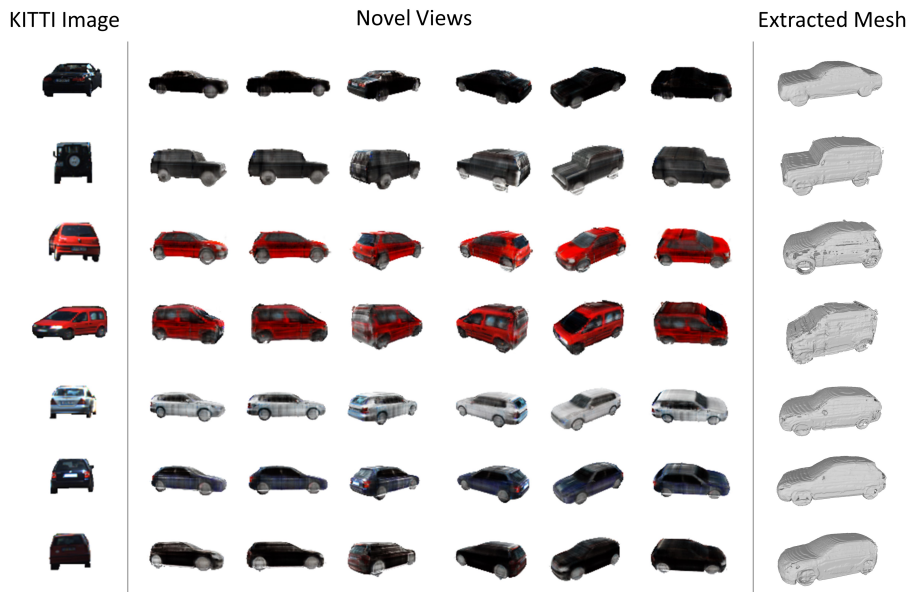

Fig. 15: Qualitatively comparing conditional NeRF generation of KITTI Cars images. We show the input in-the-wild image, rendered novel views of the generated NeRFs, and extracted meshes.

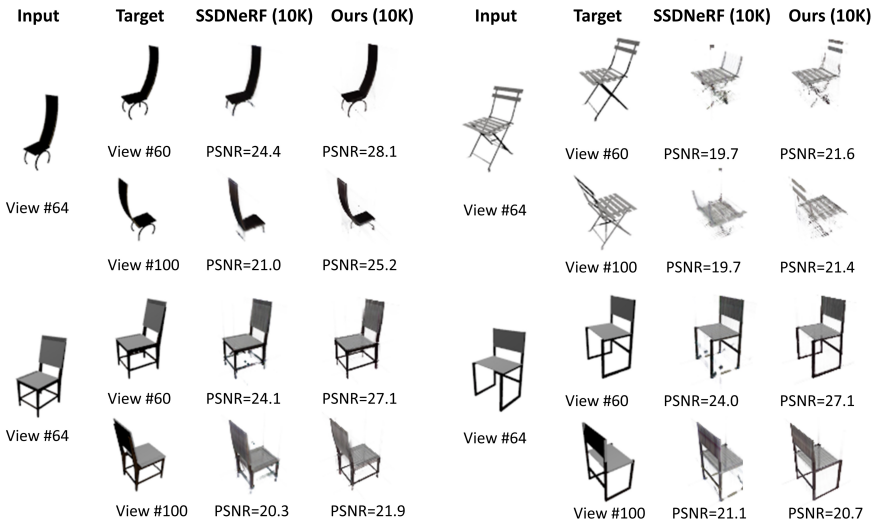

Fig. 16: Qualitatively comparing single-view NeRF reconstruction of ShapeNet test Chair images at 10,000 training iterations. Mesh2NeRF outperforms SSDNeRF, highlighting the effectiveness of our direct supervision.

References

1. Johnson, J., Ravi, N., Reizenstein, J., Novotny, D., Tulsiani, S., Lassner, C., Branson, S.: Accelerating 3d deep learning with pytorch3d. In: SIGGRAPH Asia 2020 Courses, pp. 1–1 (2020)
2. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
3. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. Advances in Neural Information Processing Systems **32** (2019)
4. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial intelligence and machine learning for multi-domain operations applications. vol. 11006, pp. 369–386. SPIE (2019)
5. Zhou, Q.Y., Park, J., Koltun, V.: Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847 (2018)