Tetra-NeRF: Representing Neural Radiance Fields Using Tetrahedra

Supplementary Material

First, in Section 1, we describe the **attached video**[1] where we illustrate the tetrahedra field and the optimisation process. Next, we extend Sections 4.2, 4.3, and 4.4 from the main paper by giving detailed results on the Blender [11], Tanks and Temples [7], and Mip-NeRF 360 [3] datasets in Sections 2, 3, and 4. Finally, we extend Section 4.5 from the main paper and evaluate how the performance changes when varying the size of the input point cloud in Section 5.

1. Attached video

To present the idea of the paper visually, we include a video illustrating the tetrahedra field representation and showing the results of the optimisation at different points in the early stages of the training. In the video, we used the *garden* scene from the Mip-NeRF 360 dataset [3]. We show the optimisation using both the sparse and the dense input point clouds. The video starts by showing the initial point cloud and the tetrahedra obtained by the Delaunay triangulation. It then shows how the scene is optimised from the first iteration. Finally, it presents the resulting video generated by the fully trained model.

2. Blender results

In Sec. 4.2 and Tab. 1 of the main paper, we presented results averaged over all scenes in the Blender dataset [11]. In the following, we present results individually per scene. The quantitative results can be seen in Table 1. We report the PSNR, SSIM, and LPIPS (VGG) [16] metrics. The evaluation protocol is the same as in Point-NeRF [14], and we use the same input point cloud as Point-NeRF^{col}. From the results, we can see that we outperform the closest approach to ours, Point-NeRF, in all three metrics on almost all scenes when using the same input point cloud (row Point-NeRF^{col}). We are slightly outperformed by Point-NeRF^{mvs}, which uses denser input point clouds obtained from its end-to-end optimised Multi-View Stereo (MVS) pipeline. The results are more pronounced on the *ficus* scene where Point-NeRF performs exceptionally well, outperforming other baselines, including Mip-NeRF [2]. Note that both Point-NeRF configurations grow the point cloud during training and, therefore, the complexity of the scene representation grows. For us, the points are fixed, and the number of parameters stays the same. In most scenes, we also outperform Plenoxels [5] which uses a sparse grid. Note that same as Point-NeRF, Plenoxels also gradually increases the representation complexity by subdividing the grid resolution at predefined training epochs. Even though Mip-NeRF [2] outperforms our approach in terms of PSNR, our method is slightly better in terms of SSIM and on par with Mip-NeRF [2] in terms of LPIPS.

We also show rendered images from all Blender scenes in Figures 1 and 2. Some artefacts can only be noticed on scenes with highly reflective surfaces – *materials*, and *drums*. By closely inspecting the produced depth maps, one can notice that sometimes the density is non-zero in large tetrahedra connecting different parts of the object. A possible cause could be the combination of the training process and the implicit bias of our model. Since the tetrahedra field uses barycentric interpolation, the features will change linearly in the tetrahedra connecting different parts of the object. However, these tetrahedra should have a density of zero everywhere except for the regions close to the vertices. For the shallow MLP, it is difficult to represent such a function, and there is not enough pressure in the optimisation process to enforce it because the error will be close to zero since the background is white, without any texture, independently of how the density is distributed in these regions.

[1]Video can be found at: https://jkulhanek.com/tetra-nerf/video.html

PSNR↑

	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*mean*
NPBG [1]	26.47	21.53	24.60	29.01	24.84	21.58	26.62	21.83	24.56
NeRF [11]	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65	31.01
NSVF [9]	33.19	25.18	31.23	37.14	32.54	32.68	34.27	27.93	31.77
Mip-NeRF [2]	37.14	27.02	33.19	39.31	35.74	32.56	38.04	33.08	34.51
Instant-NGP [12]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
Plenoxels [5]	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62	31.71
Point-NeRF^{col} [14]	35.09	25.01	33.24	35.49	32.65	26.97	35.54	30.18	31.77
Point-NeRF^{mvs} [14]	35.40	26.06	36.13	37.30	35.04	29.61	35.95	30.97	33.31
Tetra-NeRF	35.05	25.01	33.31	36.16	34.75	29.30	35.49	31.13	32.53

SSIM↑

	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*mean*
NPBG [1]	0.939	0.904	0.940	0.964	0.923	0.887	0.959	0.866	0.923
NeRF [11]	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856	0.947
NSVF [9]	0.968	0.931	0.973	0.980	0.960	0.973	0.987	0.854	0.953
Mip-NeRF [2]	0.981	0.932	0.980	0.982	0.978	0.959	0.991	0.882	0.961
Plenoxels [5]	0.977	0.933	0.890	0.985	0.976	0.975	0.980	0.949	0.958
Point-NeRF^{col} [14]	0.990	0.944	0.989	0.986	0.983	0.955	0.993	0.941	0.973
Point-NeRF^{mvs} [14]	0.991	0.954	0.993	0.991	0.988	0.971	0.994	0.942	0.978
Tetra-NeRF	0.990	0.947	0.989	0.989	0.987	0.968	0.993	0.994	0.982

LPIPS↓

	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	*mean*
NPBG [1]	0.085	0.112	0.078	0.075	0.119	0.134	0.060	0.210	0.109
NeRF [11]	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206	0.081
Mip-NeRF [2]	0.021	0.065	0.020	0.027	0.021	0.040	0.009	0.138	0.043
Plenoxels [5]	0.031	0.067	0.026	0.037	0.028	0.057	0.015	0.134	0.049
Point-NeRF^{col} [14]	0.026	0.099	0.028	0.061	0.031	0.100	0.019	0.134	0.062
Point-NeRF^{mvs} [14]	0.023	0.078	0.022	0.037	0.024	0.072	0.014	0.124	0.049
Tetra-NeRF	0.016	0.073	0.023	0.027	0.022	0.056	0.011	0.103	0.041

Table 1. **Detailed results on the Blender dataset [11].** We show the PSNR, SSIM, and LPIPS (VGG) results averaged over the testing images. We highlight the best, second, and third values. We outperform Point-NeRF^{col} [14] which was evaluated with the same input point cloud as our method.

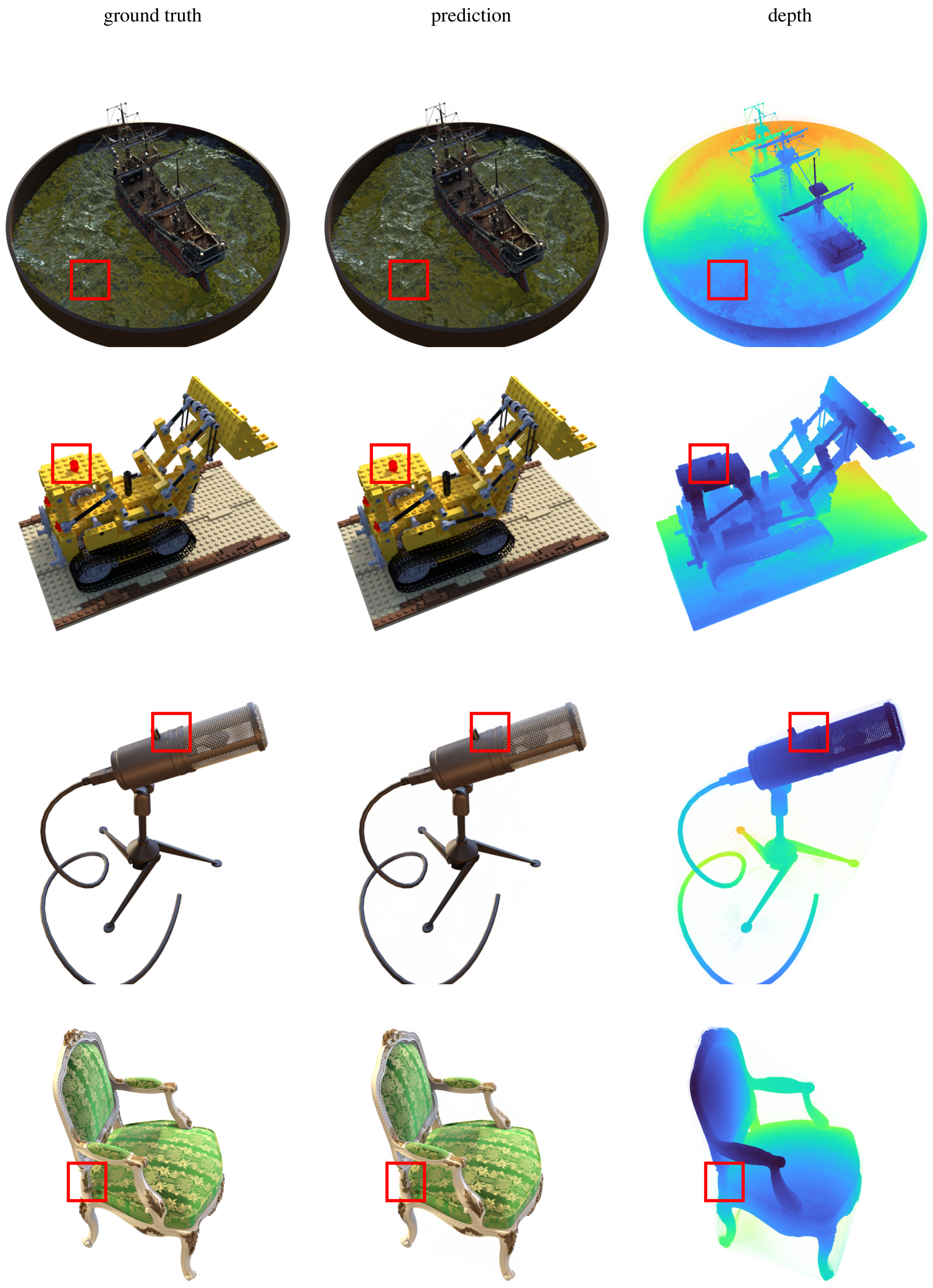

Figure 1. **Results on the Blender dataset [11] (part 1).** We show the **ground-truth** image, the **prediction**, and the **predicted depth map** on scenes: *ship* **(top)**, *lego*, *mic*, and *chair* **(bottom)**. The red squares highlight regions where there are visible errors in the images. Notice how our approach is able to recover fine textures in *chair* scene and tiny details in the *ship* scene geometry. However, the density seems to be non-zero in tetrahedra connecting different legs of the *chair*.

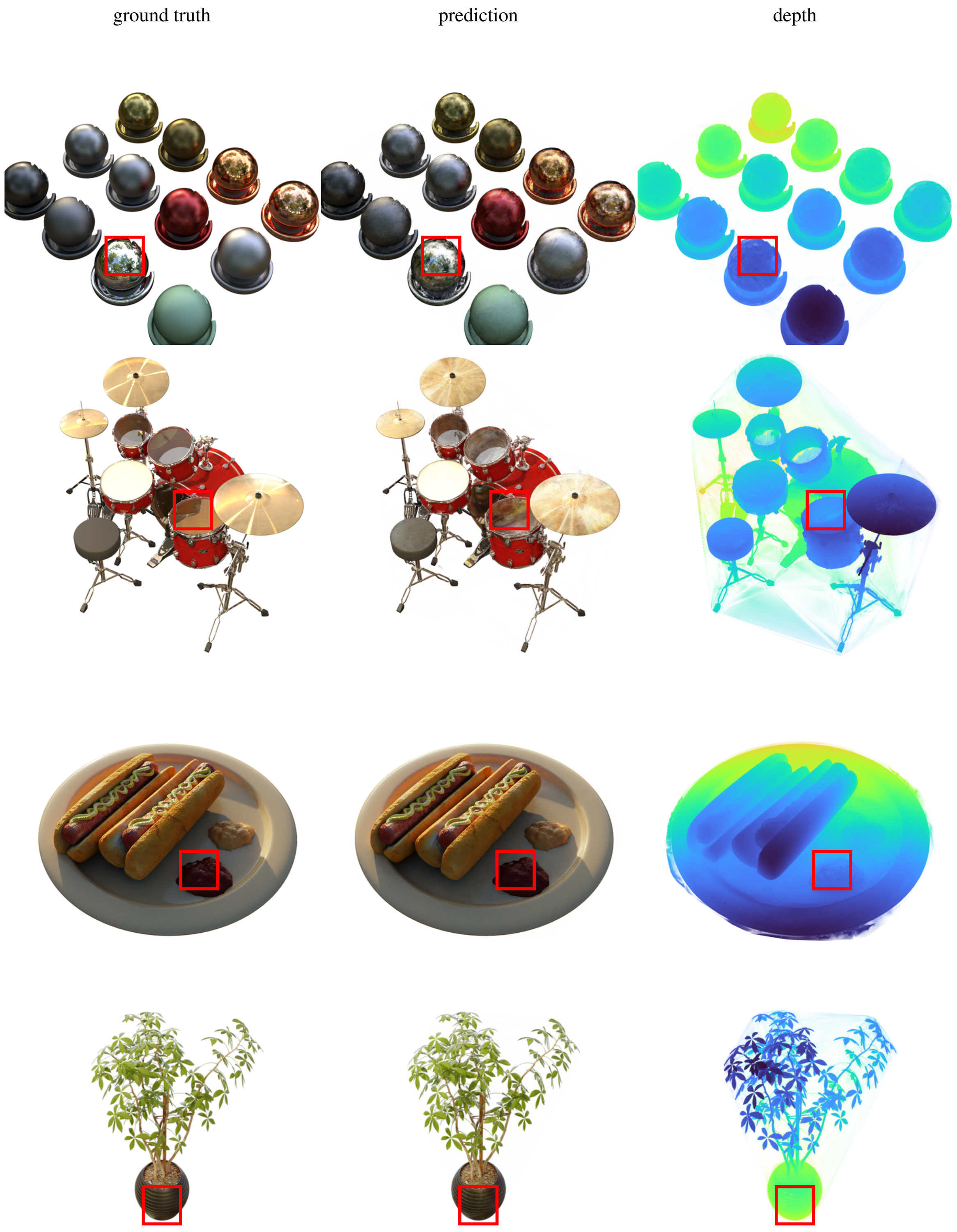

Figure 2. **Results on the Blender dataset [11] (part 2).** We show the **ground-truth** image, the **prediction**, and the **predicted depth map** on scenes: *materials* **(top)**, *drums*, *hotdog*, and *ficus* **(bottom)**. The red squares highlight regions where there are visible errors in the images. Some artefacts can be noticed in the top two scenes, where there are highly reflective surfaces. Also, the density seems to be non-zero in tetrahedra connecting distant parts of the 3D object.

3. Tanks and Temples results

We show the detailed per-scene results for the Tanks and Temples dataset [7]. Note that to be able to compare with the Point-NeRF method [14], we used the data pre-processing and splits proposed by NSVF [9], where the background is masked out. The quantitative results can be seen in Table 2. We report the PSNR, SSIM, and LPIPS (Alex) [16] metrics. The evaluation protocol is the same as in Point-NeRF [14], but we evaluate on the original resolution, the same as other compared methods. To be able to compare with Point-NeRF [14] which evaluated on a lower-resolution images[2], we have recomputed its metrics with the same full image resolution 1920×1080. To obtain the initial point clouds, we used the dense COLMAP reconstruction, which was computed using the known intrinsic and extrinsic camera parameters. However, the NSVF [9] published split had corrupted camera parameters for the *ignatius* scene, and we had to run COLMAP reconstruction from scratch to obtain both the camera poses and intrinsics. This is likely the reason for the worse results on that scene. Otherwise, we outperform all baseline methods on all metrics.

We also extend Fig. 6 from the main paper and show more qualitative results in Figure 3. Compared to Point-NeRF [14], our method produces less noisy images. When looking at the depth maps, we can observe similar non-zero density artefacts to the ones observed on the Blender dataset. Same as in the Blender dataset case, a likely cause could be a combination of the implicit bias of our method and the usage of non-textured background.

PSNR ↑

	barn	*caterpillar*	*family*	*ignatius*	*truck*	*mean*
NV [10]	20.82	20.71	28.72	26.54	21.71	23.70
NeRF [11]	24.05	23.75	30.29	25.43	25.36	25.78
NSVF [9]	27.16	26.44	33.58	27.91	26.92	28.40
Point-NeRF [14]*	27.40	25.58	33.57	28.39	26.83	28.35
Tetra-NeRF	28.86	26.64	34.27	27.17†	27.58	28.90

SSIM ↑

	barn	*caterpillar*	*family*	*ignatius*	*truck*	*mean*
NV [10]	0.721	0.819	0.916	0.992	0.793	0.848
NeRF [11]	0.750	0.860	0.932	0.920	0.860	0.864
NSVF [9]	0.823	0.900	0.954	0.930	0.895	0.900
Point-NeRF [14]*	0.908	0.927	0.976	0.959	0.939	0.942
Tetra-NeRF	0.942	0.944	0.985	0.962	0.952	0.957

LPIPS ↓

	barn	*caterpillar*	*family*	*ignatius*	*truck*	*mean*
NV [10]	0.117	0.312	0.479	0.280	0.111	0.260
NeRF [11]	0.111	0.192	0.395	0.196	0.098	0.198
NSVF [9]	0.106	0.148	0.307	0.141	0.063	0.153
Point-NeRF [14]*	0.142	0.118	0.034	0.064	0.091	0.090
Tetra-NeRF	0.087	0.077	0.021	0.050	0.062	0.059

Table 2. **Tanks and Temples results.** We show the PSNR, SSIM, and LPIPS (Alex) results averaged over the testing images. We highlight the best, second, and third values. We outperform all compared methods in all metrics on all scenes except for *ignatius*, where we did not have the correct camera parameters and had to run camera pose estimation prior to training.

[2] https://github.com/Xharlie/pointnerf/issues/62

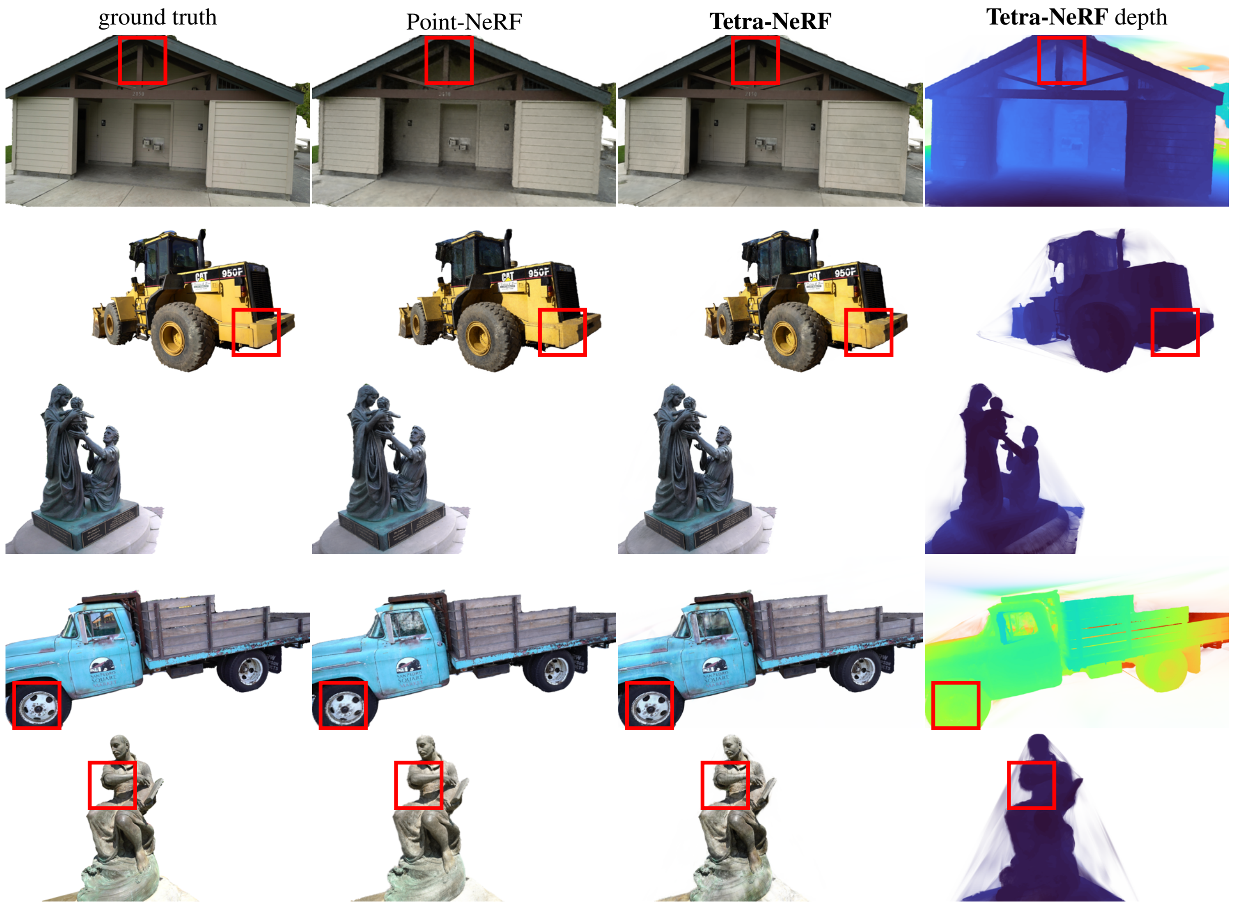

Figure 3. **Results on the Tanks and Temples dataset [7].** We show the **ground-truth** image, the **prediction**, and the **predicted depth map** on scenes: *barn* **(top)**, *caterpillar*, *family*, *truck*, and *ignatius* **(bottom)**. We also show a comparison with Point-NeRF [14]. The red squares highlight regions where our method has fewer artefacts compared to Point-NeRF. Notice how our method produces less noisy images compared to Point-NeRF. We can observe similar non-zero density artefacts to the ones observed on the Blender dataset. Again, we attribute it to a combination of implicit bias and non-textured background.

4. Mip-NeRF 360 results

We also extend the results on the Mip-NeRF 360 dataset [3], presented in Table 8 and Figure 4 in the main paper, by showing detailed results for each scene. We follow the same training and evaluation procedure as Mip-NeRF 360 [3], and for the outdoor and indoor scenes, we train and evaluate with 4x and 2x downsampled images, respectively. The PSNR, SSIM, and LPIPS (Alex) [16] results are presented in Table 3. In terms of SSIM, our approach performs slightly worse than Point-Based Neural Rendering [8], Stable View Synthesis [13], and Mip-NeRF 360 [3]. In terms of the PSNR and LPIPS metrics, our approach performs comparable or better than these baselines. *E.g.*, the state-of-the-art Mip-NeRF 360 [3] has a slightly better PSNR and Tetra-NeRF has a slightly better LPIPS. We typically outperform Stable View Synthesis [13] and competitors other than Mip-NeRF 360 in terms of PSNR. Note that similarly to Tetra-NeRF, Stable View Synthesis uses a geometric prior as input – a mesh instead of a point cloud.

We also show qualitative results for indoor and outdoor Mip-NeRF 360 scenes in Figures 4 and 5. We notice more artefacts in the outdoor scenes compared to indoor ones. Small, high-frequency details such as grass are not represented well, which can be visible, *e.g.*, in the *flowers* scene. One potential cause for this behaviour, which we plan to investigate in future work, could be that the poses estimated outdoors (where the camera is typically farther away from the scene than indoors) are noisier which makes it harder to recover fine details. For the indoor scenes, our model is able to represent the scenes with

PSNR

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [4, 11]	21.76	19.40	23.11	21.73	21.28	28.56	25.67	26.31	26.81
mip-NeRF [2]	21.69	19.31	23.16	23.10	21.21	28.73	25.59	26.47	27.13
NeRF++ [15]	22.64	20.31	24.32	24.34	22.20	28.87	26.38	27.80	29.15
Deep Blending [6]	21.09	18.13	23.61	24.08	20.80	27.20	26.28	25.02	27.08
Point-Based Neural Rendering [8]	21.64	19.28	22.50	23.90	20.98	26.99	25.23	24.47	28.42
Stable View Synthesis [13]	22.79	20.15	25.99	24.39	21.72	28.93	26.40	28.49	29.07
mip-NeRF 360 [3]	23.95	21.60	25.09	25.98	21.99	28.24	28.40	30.81	30.27
Tetra-NeRF	23.53	20.36	26.15	24.42	21.41	32.02	28.02	29.66	31.13

SSIM

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [4, 11]	0.455	0.376	0.546	0.453	0.459	0.843	0.775	0.749	0.792
mip-NeRF [2]	0.454	0.373	0.543	0.517	0.466	0.851	0.779	0.745	0.818
NeRF++ [15]	0.526	0.453	0.635	0.594	0.530	0.852	0.802	0.816	0.876
Deep Blending [6]	0.466	0.320	0.675	0.634	0.523	0.868	0.856	0.768	0.883
Point-Based Neural Rendering [8]	0.608	0.487	0.735	0.651	0.579	0.887	0.868	0.876	0.919
Stable View Synthesis [13]	0.663	0.541	0.818	0.683	0.606	0.905	0.886	0.910	0.925
mip-NeRF 360 [3]	0.687	0.582	0.800	0.745	0.619	0.907	0.890	0.916	0.932
Tetra-NeRF	0.614	0.470	0.775	0.613	0.456	0.894	0.850	0.877	0.905

LPIPS

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [4, 11]	0.536	0.529	0.415	0.551	0.546	0.353	0.394	0.335	0.398
mip-NeRF [2]	0.541	0.535	0.422	0.490	0.538	0.346	0.390	0.336	0.370
NeRF++ [15]	0.455	0.466	0.331	0.416	0.466	0.335	0.351	0.260	0.291
Deep Blending [6]	0.377	0.476	0.231	0.351	0.383	0.266	0.258	0.246	0.275
Point-Based Neural Rendering [8]	0.313	0.372	0.197	0.303	0.325	0.216	0.209	0.160	0.178
Stable View Synthesis [13]	0.243	0.317	0.137	0.281	0.286	0.182	0.168	0.125	0.164
mip-NeRF 360 [3]	0.296	0.343	0.173	0.258	0.338	0.208	0.206	0.129	0.182
Tetra-NeRF	0.271	0.378	0.136	0.274	0.429	0.104	0.127	0.098	0.084

Table 3. **Detailed Mip-NeRF 360 [3] results.** We show the PSNR, SSIM, and LPIPS (Alex) results averaged over the testing images. We highlight the best, second, and third values. Our method has a worse SSIM. However, Tetra-NeRF seems to perform comparably to Mip-NeRF 360 [3] on most scenes in terms of PSNR and LPIPS, where we seem to achieve a slightly higher LPIPS and a slightly lower PSNR. We also outperform Stable View Synthesis [13] and all competitors other than Mip-NeRF 360 on all scenes in terms of PSNR.

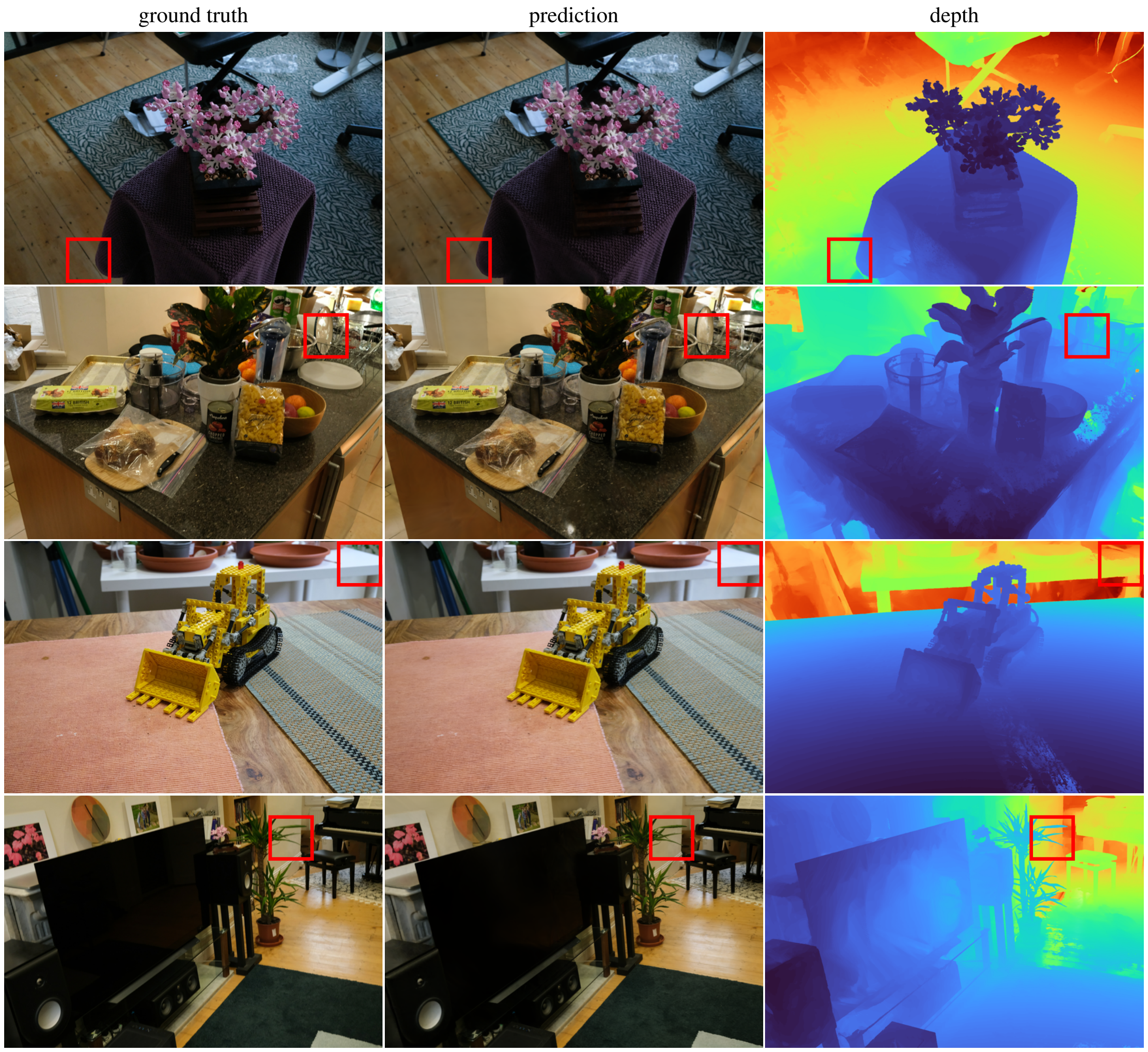

Figure 4. **Results on the indoor scenes from Mip-NeRF 360 dataset [3].** We show the **ground-truth** image, the **prediction**, and the **predicted depth map** on scenes: *bonsai* **(top)**, *counter*, *kitchen*, and *room* **(bottom)**. The red squares highlight regions where there are visible errors in the images. Tetra-NeRF is able to represent even fine details, such as texts on products in the *counter* scene, well.

very high fidelity, including very fine details such as texts on products in the *counter* scene.

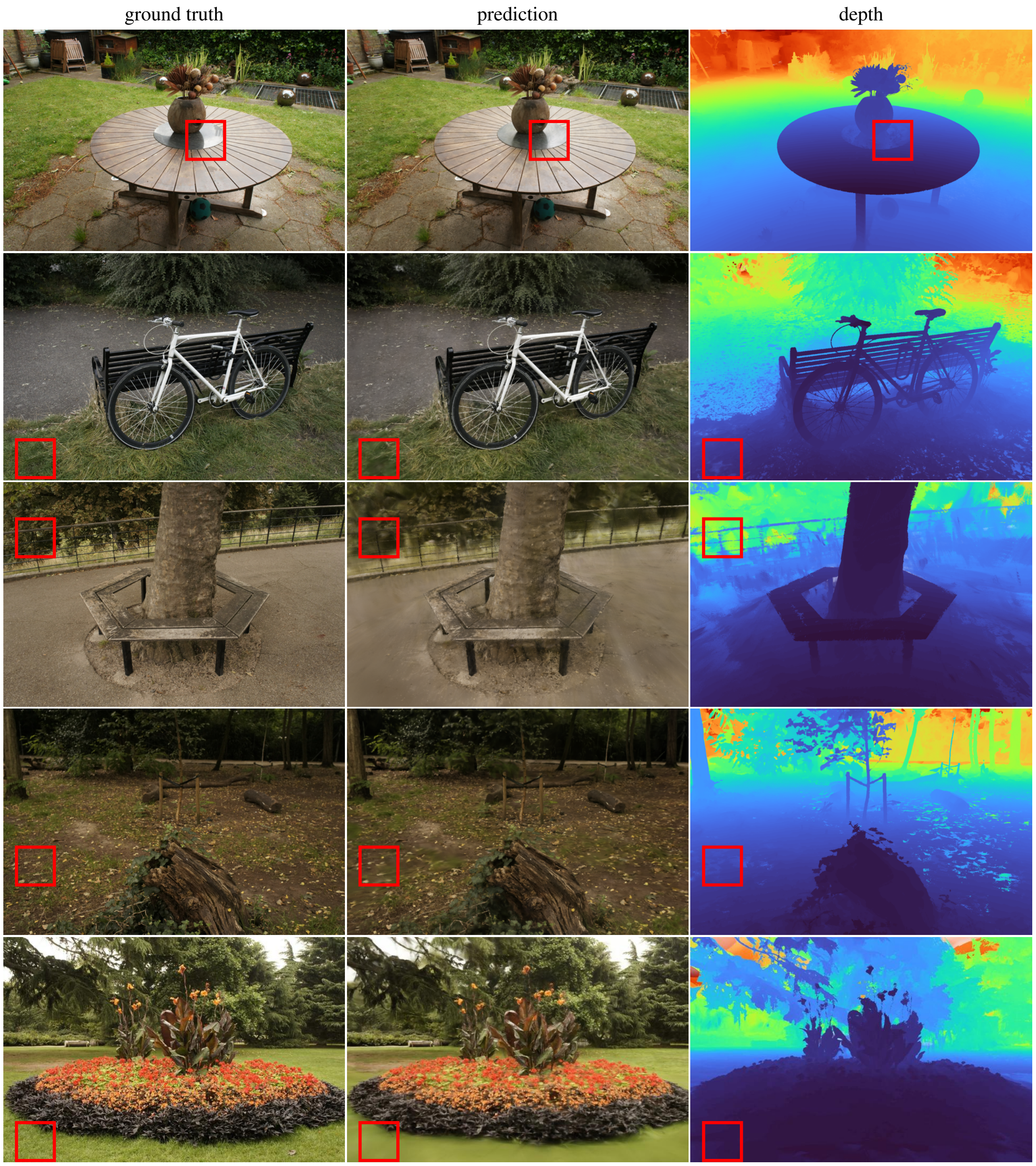

Figure 5. **Results on the outdoor scenes from Mip-NeRF 360 dataset [3].** We show the **ground-truth** image, the **prediction**, and the **predicted depth map** on scenes: *garden* **(top)**, *bicycle*, *treehill*, *stump*, and *flowers* **(bottom)**. The red squares highlight regions where there are visible errors in the images. Tetra-NeRF is not able to represent the grass and the ground in *flowers* and *treehill* scenes well, and we can see blur artefacts. On the *garden* scene, the reconstruction achieves high fidelity except for the centre of the table, where the reflections are slightly incorrect.

5. Varying the number of input points

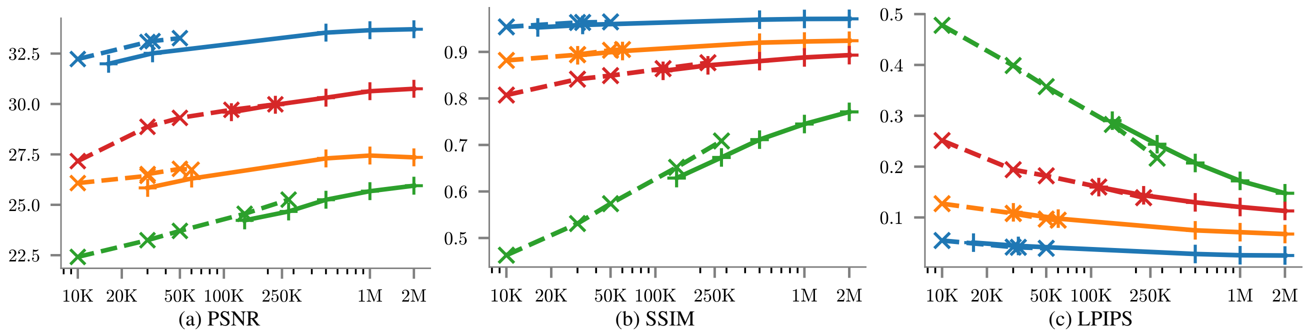

Figure 6. **Performance with different sizes and quality of the input point cloud. a, b, c)** shows the PSNR, SSIM, and LPIPS [16] with different sizes of input point clouds (PSNR is repeated from the main paper). The solid and dashed lines represent the dense and coarse COLMAP reconstructions. The following scenes were evaluated (best to worst): *tt/family*, *360/room*, *tt/truck*, and *360/garden* from the Tanks and Temples [7] (tt) and the Mip-NeRF 360 [3] (360-indoor/360-outdoor) datasets. As expected, the quality increases with the number of points, but also sparse reconstruction performs better at the same number of points.

We conducted a study on the effect of different sizes of the input point cloud. In the main paper, we presented the PSNR results on four scenes. Here we also show the SSIM and LPIPS. We consider both the sparse and dense COLMAP reconstructions and analyse the performance as we sub-sample the points or add more points randomly. In order to save computational resources, we only train the method for 100k iterations. The results are visualised in Figure 6.

As expected, the performance improves with the number of points used, as it leads to a finer subdivision of the scene around the surface. Note how the performance of dense reconstruction is lower than sparse reconstruction with the same size. Also, note how the performance of *360/garden* increases when adding randomly sampled points to the sparse point cloud. For Tetra-NeRF it is important to have a dense-enough coverage in regions close to surfaces and randomly subsampling a larger point cloud may miss some regions. Similarly, adding more points at random in proximity to existing ones helps as it increases the density in those regions. For all experiments in the paper and the *Supp. Mat.*, we used random sub-sampling. An interesting direction for future work is to investigate more sophisticated strategies that, *e.g.*, sample more points around fine details such as corners and edges and fewer points in planar regions.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 2

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 1, 2, 7

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 7, 8, 9, 10

[4] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 7

[5] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2

[6] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *SIGGRAPH, ToG*, 37(6):1–15, 2018. 7

[7] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *SIGGRAPH, ToG*, 2017. 1, 5, 6, 10

[8] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40, pages 29–43. Wiley Online Library, 2021. 7

[9] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2, 5

[10] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *SIGGRAPH, ToG*, 38(4):1–14, 2019. 5

[11] 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 *ECCV*, 2020. 1, 2, 3, 4, 5, 7

[12] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH, ToG*, 2022. 2

[13] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 7

[14] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *CVPR*, 2022. 1, 2, 5, 6

[15] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 7

[16] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, June 2018. 1, 5, 7, 10