Supplementary Materials for NeuManifold

Xinyue Wei[1], Fanbo Xiang[1], Sai Bi[2], Anpei Chen[3,4], Kalyan Sunkavalli[2]
Zexiang Xu[2*], Hao Su[1*]

[1]University of California San Diego, [2]Adobe Research, [3]ETH Zürich, [4]University of Tübingen

1. Prelimiaries

1.1. Watertight and Manifold Meshes

Watertight. If all edges are shared by exactly two faces, then the mesh is watertight.

Manifold. A manifold mesh must meet the following properties: (1) all edges must connect at most two faces; (2) each edge is incident to one or two faces and faces incident to a vertex must form a closed or open fan; (3) the faces must not self-intersect with each other.

1.2. Volumetric Neural Fields

Recent neural field representations utilize differentiable volume rendering for their reconstruction and leads to high visual quality. While our approach can generally support any neural field models, we apply TensoRF and NeuS in our paper. We now briefly cover the preliminaries of the method.

The original NeRF uses pure MLPs, which make it slow to train and incapable of modeling details accurately. TensoRF [2] decodes the radiance field from a volume of features, and this feature volume is further factorized into factors leveraging CANDECOMP/PARAFAC decomposition or vector-matrix decomposition. In this work, we are interested in the vector-matrix decomposition, which factorizes the 4D feature volume as the sum of three outer products between a matrix and a vector.

1.3. Differentiable Rasterization

Differentiable rasterization refers to methods that optimize inputs of rasterization-based rendering pipelines. In this work, we are interested in nvdiffrast [4], which consists of 4 stages, rasterization, interpolation, texture lookup, and anti-aliasing. We mainly use the rasterization stage, which maps triangles from 3D space onto pixel space, and the interpolation stage, which provides 3D coordinates of pixels to query the appearance network.

[1]Research partially done when X. Wei was an intern at Adobe Research
[2]* Equal advisory.

To ensure the mesh optimized by differentiable rasterization is a watertight manifold, we need to apply a meshing algorithm that generates such meshes. In this work we propose DiffMC, which divides the 3D space into a deformable grid and takes a scalar field (often SDF) defined on its vertices as input. The algorithm turns the scalar field into an explicit mesh by a differentiable marching cubes algorithm.

2. Implementation Details

For the first stage, we directly build on off-the-shelf volume rendering models. Specifically, for TensoRF, we use the official implementation. We compare two of our models for our main results: a high-quality one, labeled with Ours, which uses the TensoRF (VM) with 48-dim input features and 12-dim output features, plus a three-layer MLP decoder; a fast one, labeled with Ours-F that uses the TensoRF (VM) with 48-dim input features and output 27-dim SH coefficients.

To adapt the density values for DiffMC, we transform these values into opacity using the formula: $\alpha = 1 - \exp(-\sigma \cdot \delta)$, where σ represents density, α denotes opacity, and δ is the ray step size used in volume rendering. We employ a threshold t to control the surface's position relative to opacity and use the value $\alpha - t$ for mesh extraction.

We train all the stage 2 and 3 models with batch size of 2 for 10k iterations. We use DiffMC with a grid resolution of 256 for all results. Except when comparing with nvdiffrec, we use the default resolution of 128 as nvdiffrec's performance drops on higher resolutions, possibly due to the decreased batch size and harder optimization.

3. Differentiable Marching Cubes (DiffMC)

In this section, we present additional results for DiffMC. These include a 2D example showing why DMTet tends to introduce more artifacts on density fields than DiffMC, an ablation study that demonstrates how grid resolution influences visual fidelity and a comparison highlighting the effectiveness of our method in mesh reconstruction when compared to DMTet [8].

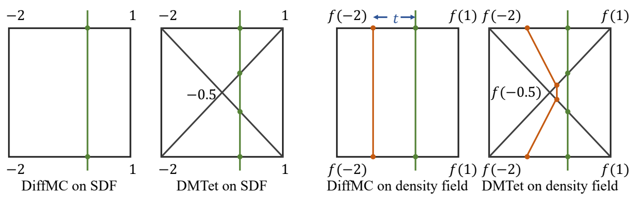

Figure 1. 2D example illustrating why DMTet tends to introduce more artifacts when extracting meshes from density fields while DiffMC can generate much smoother surfaces.

First, we illustrate how DMTet and DiffMC generate surfaces with a 2D schematic diagram in Fig. 1. In 2D, Marching Cubes is analogous to "Marching Squares" and Marching Tetrahedra is analogous to "Marching Triangles". Given a surface (shown as a green vertical line) passing through the square/triangle grids (shown as black lines), suppose we have recorded the perfect signed distance function (SDF) values of the surface on the grid nodes, as shown in the two leftmost figures, regardless of how the algorithm divides the space, both methods exactly recover the ground truth surface through linear interpolation.

However, in practice, perfect SDF values are not easily obtainable, especially when the input comes from a volumetric density representation. Here, we simulate an imperfect SDF by applying a non-linear transformation $f(s) = \exp(s) - 1 - t$ to the SDF values. Under this scenario, DiffMC can still generate a flat surface (red line in the second figure from the right), albeit with a slight offset t which can be rectified by introducing an adjustable threshold to the grid values. In contrast, DMTet produces zigzag lines (red line in the rightmost figure) due to varying space divisions and cannot be easily fixed.

As we transition from lower to higher resolutions, we observe a consistent improvement in rendering quality, ultimately converging as the resolution reaches 400, as demonstrated in Tab. 1. Moreover, as depicted in Fig. 2, a higher-resolution DiffMC is notably more adept at recovering intricate structures, such as the ropes on the ship.

Next, we highlight the advantages of our method in extracting meshes from density fields by applying both our approach and DMTet [8] to a set of pre-trained density networks, including TensoRF [2], instant-NGP [7] and vanilla NeRF [6]. By comparing the visible surface agreemen (VSA) of the reconstructed meshes, as illustrated in Fig. 3, we observe a consistent enhancement brought about by DiffMC across all methods. We also conduct a comparison between our DiffMC and DMTet in our pipeline, noting a significant improvement in surface smoothness with our method, which effectively mitigates most of the artifacts resulting from the non-linearity of the density field.

4. Mip-NeRF 360 Dataset

We evaluate our method on unbounded real scenes in the Mip-NeRF 360 dataset [1]. To deal with the unbounded background, we follow the contraction function proposed in [1] to warp the far objects from their original space, t-space, into the contracted space s-space (a sphere with a radius of 1.2 in our setup). When generating the mesh, we apply DiffMC on the geometry network within t-space so that the mesh can be watertight manifold, otherwise the contraction may break the property. After getting the points on the mesh surface, we contract the points back to s-space to compute the color. Within the t-space, we utilize multiple resolutions for the entire scene, with a higher resolution (340) for the foreground and a lower resolution (56) for the background. To represent the distant background that falls outside the [-4, 4] box range, we employ a skybox. We use the anti-aliasing of nvdiffrast [4] for this dataset.

Our method generates watertight manifold foreground meshes. Therefore, we can apply simulation algorithms on the foreground objects, as shown in Fig. 7, where we apply soft-body simulation on the flower and use a solid ball to hit it.

In Tab. 2, we compared our method with others. Some mesh rendering methods, such as MobileNeRF [3] and nerf2mesh [9], provided results for selected scenes, while our method worked effectively on all unbounded scenes, particularly excelling in indoor scenes.

5. LLFF dataset

We evaluate our method on forward-facing scenes on LLFF dataset [5]. Following [2], we contract the whole scene into NDC space to do the reconstruction and mesh extraction. On this dataset, we use DiffMC with resolution of 375. We use 9× sample per-pixel SSAA for this dataset. Tab. 3 and Fig. 9 shows the quantitative and qualitative results. Fig. 10 shows the reconstructed mesh of the scenes.

We put our method to the test with forward-facing scenes from the LLFF dataset [5]. In line with [2], we condensed the entire scene into NDC space for reconstruction and mesh extraction. For this dataset, we employed DiffMC with a resolution of 375. You can find both the quantitative results in Tab. 3 and the qualitative results in Fig. 9. Additionally, Fig. 10 showcases the reconstructed mesh for these scenes.

6. NeRF-Synthetic Dataset

We show the complete quantitative comparison between our method and the previous works on the NeRF-Synthetic dataset in Tab. 4 and the complete visual comparison in Fig. 12.

Table 1. The influence of DiffMC resolution to rendering quality. The visual fidelity consistently improves as the resolution increases, eventually reaching a plateau when it reaches 400.

DiffMC reso	32	64	100	128	200	256	300	384	400
PSNR	23.12	26.83	28.64	29.46	30.8	31.19	31.34	31.53	31.54
SSIM	0.894	0.925	0.94	0.946	0.952	0.954	0.955	0.956	0.956
LPIPS	0.121	0.089	0.075	0.069	0.061	0.059	0.057	0.056	0.056

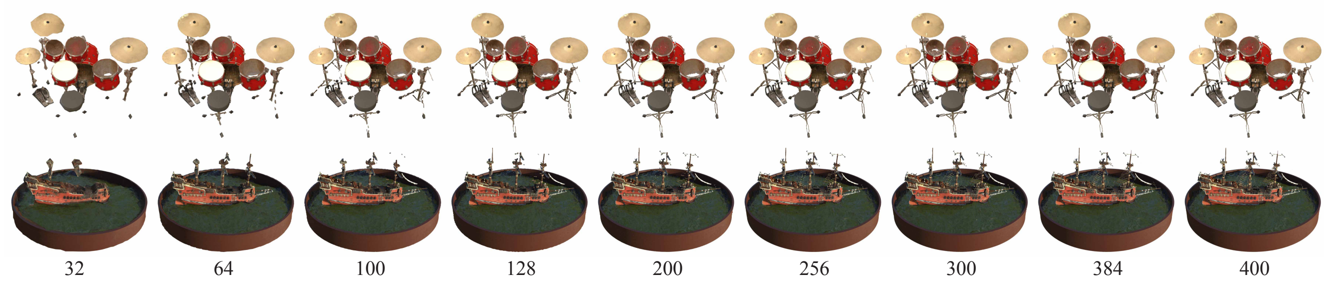

Figure 2. The influence of DiffMC resolution to rendeirng quality. We have noticed that lower resolutions can capture most of the coarse structures but tend to lose finer details, such as the drum legs and the ropes on the ship. These finer details become more discernible as the resolution increases.

Table 2. Quantitative results on each scene in the Mip-NeRF 360 dataset.

PSNR	Bicycle	Garden	Stump	Flowers	Treehill	Bonsai	Counter	Kitchen	Room	Mean
MobileNeRF	**21.70**	23.54	**23.95**	**18.86**	**21.72**	-	-	-	-	-
nerf2mesh	22.16	22.39	22.53	-	-	-	-	-	-	-
BakedSDF	-	-	-	-	-	-	-	-	-	24.51
Ours (HQ-m)	20.16	23.36	22.27	18.49	21.07	26.64	24.83	24.97	26.75	23.17
Ours (HQ)	21.38	**24.90**	23.51	18.82	21.64	**28.61**	**26.31**	**26.63**	**28.95**	**24.53**
SSIM										
MobileNeRF	0.426	0.599	0.556	0.321	0.450	-	-	-	-	-
nerf2mesh	**0.470**	0.500	0.508	-	-	-	-	-	-	-
BakedSDF	-	-	-	-	-	-	-	-	-	**0.697**
Ours (HQ-m)	0.382	0.616	0.492	0.334	0.447	0.835	0.746	0.644	0.815	0.590
Ours (HQ)	0.469	**0.746**	**0.589**	**0.366**	**0.494**	**0.888**	**0.808**	**0.764**	**0.872**	0.666
LPIPS										
MobileNeRF	0.513	0.358	0.430	0.526	0.522	-	-	-	-	-
nerf2mesh	0.510	0.434	0.490	-	-	-	-	-	-	-
BakedSDF	-	-	-	-	-	-	-	-	-	**0.309**
Ours (HQ-m)	0.561	0.372	0.475	0.553	0.560	0.268	0.346	0.380	0.348	0.429
Ours (HQ)	**0.488**	**0.252**	**0.413**	**0.520**	**0.506**	**0.201**	**0.270**	**0.275**	**0.274**	0.355

7. Mesh Quality

We show the mesh quality comparison in Fig.13, where except for Mobile-NeRF [3] and nerf2mesh [9], all the meshes are watertight manifold. We show the VSA-tolerance curves for the scenes in NeRF-Synthetic in Fig. 11.

8. Network Architecture

In this section, we describe the network architecture used in the experiments. Our proposed method has two versions, a high-quality one and a fast one, and they share the same geometry network architecture but with different appearance networks. The geometry network is the same as TensoRF [2] VM-192 in its paper. The appearance network is from TensoRF and we show the two versions below respectively.

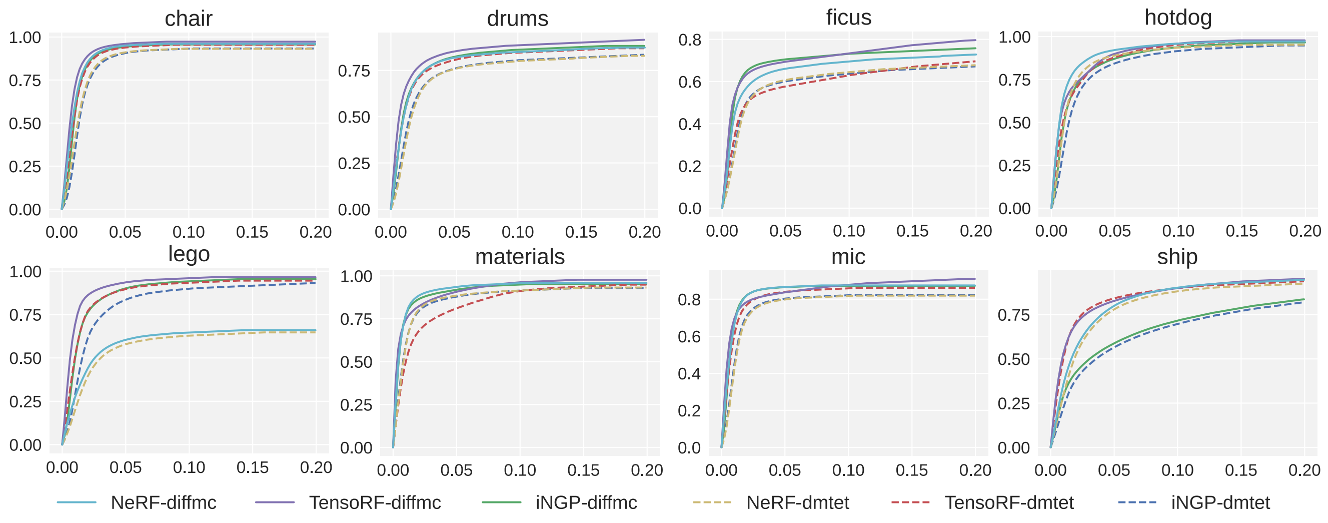

Figure 3. DMTet vs DiffMC on extracting meshes from pre-trained density fields. Across all three methods, DiffMC consistently outperforms DMTet in terms of mesh quality.

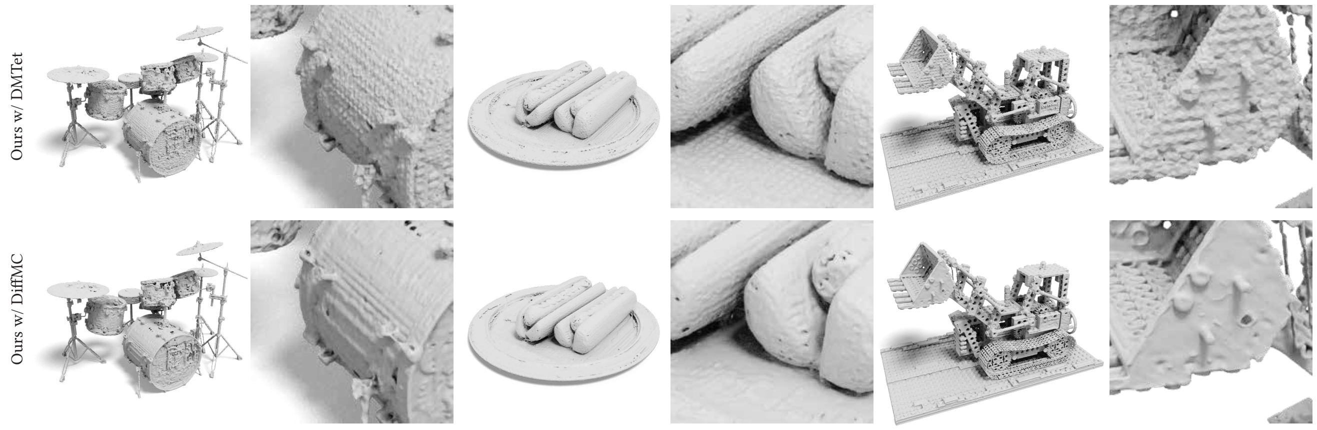

Figure 4. A mesh surface comparison of Ours between using DiffMC and DMTet reveals that DiffMC can create significantly smoother surfaces. This improvement is not limited to axis-aligned surfaces; it consistently outperforms DMTet on various rounded surfaces as well.

High-quality. We use the Vector-Matrix (VM) decomposition in TensoRF, which factorizes a tensor into multiple vectors and matrices along the axes as in Equation 3 of the TensoRF paper. The feature $\mathcal{G}_c(\mathbf{x})$ generated by VM decomposition is concatenated with the viewing direction d and put into the MLP decoder S for the output color c:

$$c = S(\mathcal{G}_c(\mathbf{x}), d), \tag{1}$$

We also apply frequency encodings (with Sin and Cos functions) on both the features $\mathcal{G}_c(\mathbf{x})$ and the viewing direction d. We use a 300^3 dense grid to represent the scenes in NeRF-Synthetic and use 2 frequencies for features and 6 frequencies for the viewing direction. The detailed network architecture is shown in Tab. 5. As for Mip-NeRF 360 and LLFF datasets we use a 512^3 dense grid to represent the unbounded indoor scenes and do not use frequency encodings.

Fast. The fast version shares similar architecture and positional encoding setups with the high-quality version before the MLP decoder but uses the spherical harmonics (SH) function as $\mathcal{G}_c$ instead, as shown in Tab. 5.

Quality Speed Trade-off We also show the model rendering quality and speed after deployment in Tab. 6.

9. Visualization of ablation Study on Stage 1

We visualize the rendering results of models using different initialization strategies during stage 1, as shown in Fig. 8. The comparison shows that employing high-resolution DiffMC grids without proper geometry initialization can lead the mesh optimization process to become stuck in suboptimal geometric configurations.

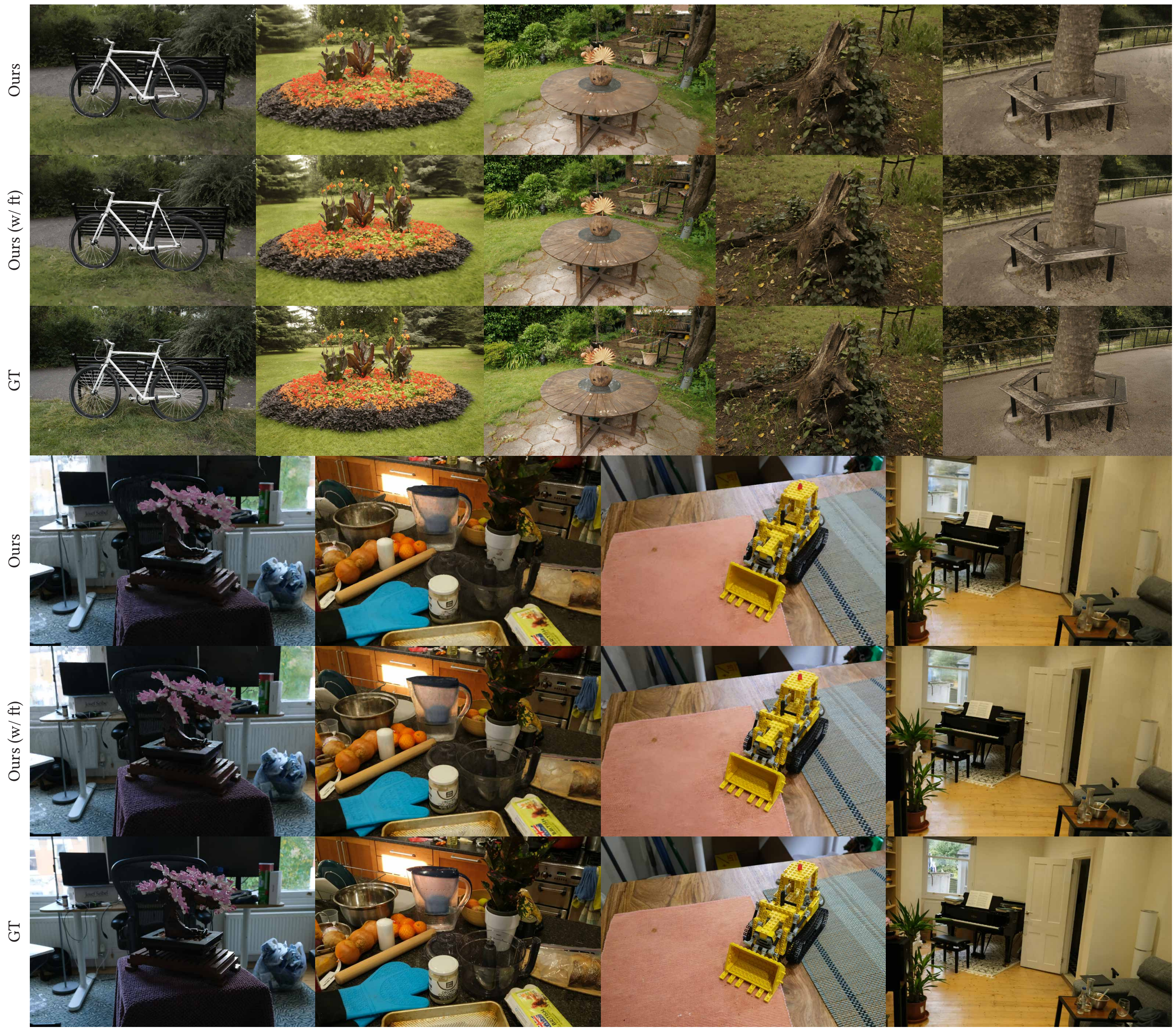

Figure 5. Mip-NeRF 360 renderings.

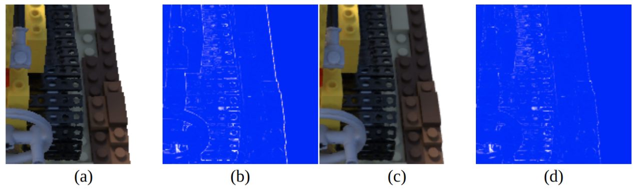

Figure 6. Comparison between 8x MSAA and no AA. (a) Our deployed high-quality model without AA (FPS: 146, PSNR: 31.26). (c) the same model with 8× MSAA (FPS: 93, PSNR: 33.01). (b) and (d) show the error maps of (a) and (c) respectively. The visual quality at edges is significantly improved by MSAA with a relatively small performance hit.

10. Ablation Study on Appearance Network

We validate the necessity of optimizing the meshes in Tab. 7. To achieve this, we compare against baselines that keep the meshes from Stage 1 fixed and only optimize the appearance. We also provide the results using the GT mesh in combination with the TensorF appearance network as a reference, representing the upper limit of texture optimization methods. As we can see from the results, TensoRF appearance network achieves the best performance. All appearance networks were trained from scratch for fair comparison.

Table 3. Quantitative results on each scene in the LLFF dataset.

PSNR	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex	Mean
MobileNeRF	**24.59**	27.05	30.82	27.09	20.54	19.66	**31.28**	26.26	25.91
nerf2mesh	23.94	26.48	28.02	26.25	19.22	19.08	29.24	25.80	24.75
Ours (F-m)	23.72	27.05	30.88	27.01	19.68	18.43	30.33	25.03	25.27
Ours (F)	24.05	**27.22**	30.98	27.09	19.92	18.91	30.63	25.58	25.55
Ours (HQ-m)	24.19	26.99	31.18	27.35	20.49	19.68	30.79	26.61	25.91
Ours (HQ)	24.54	27.08	**31.32**	**27.49**	**20.59**	**19.73**	31.11	**27.16**	**26.13**
SSIM									
MobileNeRF	**0.808**	0.839	0.891	0.864	0.711	0.647	**0.943**	0.900	0.825
nerf2mesh	0.751	**0.879**	0.765	0.819	0.644	0.602	0.914	0.868	0.780
Ours (F-m)	0.757	0.842	0.895	0.864	0.681	0.601	0.923	0.865	0.803
Ours (F)	0.772	0.848	0.898	0.866	0.693	0.622	0.926	0.872	0.812
Ours (HQ-m)	0.789	0.852	**0.902**	0.877	0.739	0.677	0.930	0.896	0.833
Ours (HQ)	0.801	0.856	**0.902**	**0.881**	**0.745**	**0.681**	0.933	**0.904**	**0.838**
LPIPS									
MobileNeRF	**0.202**	0.163	**0.115**	0.169	0.245	0.277	**0.143**	**0.147**	**0.183**
nerf2mesh	0.303	0.204	0.270	0.260	0.321	0.314	0.246	0.215	0.267
Ours (F-m)	0.274	0.181	0.158	0.196	0.254	0.278	0.208	0.256	0.226
Ours (F)	0.258	0.175	0.152	0.191	0.244	0.260	0.203	0.247	0.216
Ours (HQ-m)	0.245	0.164	0.137	0.171	0.202	0.234	0.188	0.216	0.195
Ours (HQ)	0.228	**0.160**	0.136	**0.165**	**0.198**	**0.226**	0.181	0.205	0.187

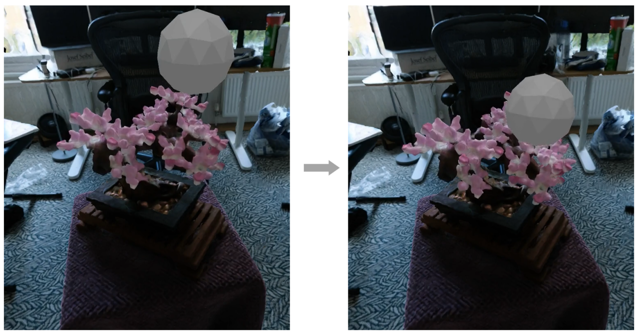

Figure 7. Soft-body simulation on the foreground watertight manifold mesh. The solid ball hits the flower and makes it deform. See the attached video.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. 1, 2, 3

[3] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 2, 3

[4] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 1, 2

[5] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[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. *Communications of the ACM*, 65(1):99–106, 2021. 2, 12

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2, 12

[8] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 1, 2

[9] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*, 2023. 2, 3

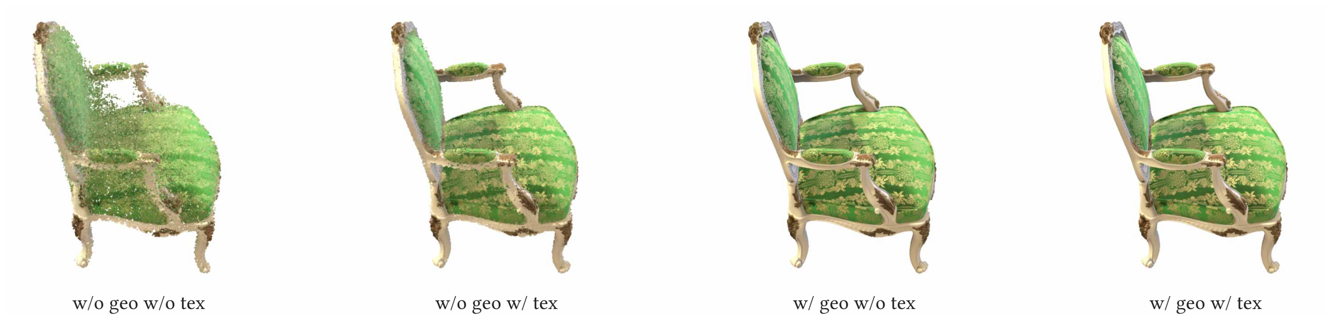

Figure 8. Visual comparison of Ours w/ or w/o geometry and texture initialization. when both initializations are omitted, the mesh optimization process can easily become trapped in local minima, as illustrated in the first left image. Although texture initialization can provide some assistance to the optimization process, it still falls short of achieving satisfactory geometric quality.

Table 4. Quantitative results on each scene in the NeRF-Synthetic dataset.

PSNR	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
MobileNeRF	34.09	25.02	30.20	35.46	34.18	26.72	32.48	29.06	30.90
nvdiffrec	31.00	24.39	29.86	33.27	29.61	26.64	30.37	26.05	28.90
TensoRF (DT)	27.72	22.20	25.66	28.85	25.86	22.12	26.13	23.67	25.28
NeuS (DT)	31.80	22.52	23.44	33.86	28.07	26.68	31.42	25.02	27.85
nerf2mesh	31.93	24.80	29.81	34.11	32.07	25.45	31.25	28.69	29.76
nvdiffrec (m)	31.24	23.17	25.11	32.67	28.44	26.33	29.39	24.82	27.65
Ours (F)	33.82	25.25	31.28	35.43	34.40	26.83	32.37	28.13	30.94
Ours (HQ)	**34.46**	**25.42**	**31.83**	**36.45**	**35.40**	**27.38**	**33.46**	**28.77**	**31.65**
Ours (F-m)	33.68	24.98	30.23	35.10	33.39	26.61	32.21	27.54	30.47
Ours (HQ-m)	34.37	25.17	30.64	36.35	34.28	27.22	33.35	28.12	31.19
SSIM									
MobileNeRF	0.978	0.927	0.965	0.973	0.975	0.913	0.979	0.867	0.947
nvdiffrec	0.965	0.921	0.969	0.973	0.952	0.924	0.975	0.827	0.938
TensoRF (DT)	0.922	0.872	0.933	0.916	0.893	0.835	0.936	0.780	0.886
NeuS (DT)	0.975	0.907	0.934	0.975	0.949	0.921	0.981	0.840	0.935
nerf2mesh	0.964	0.927	0.967	0.970	0.957	0.896	0.974	0.865	0.940
nvdiffrec (m)	0.970	0.915	0.937	0.973	0.943	0.927	0.975	0.820	0.932
Ours (F)	0.977	0.935	0.974	0.978	0.978	0.925	0.981	0.865	0.952
Ours (HQ)	**0.981**	**0.939**	**0.977**	**0.981**	**0.982**	**0.930**	**0.986**	**0.877**	**0.956**
Ours (F-m)	0.976	0.932	0.970	0.978	0.976	0.923	0.980	0.859	0.949
Ours (HQ-m)	**0.981**	0.935	0.973	**0.981**	0.979	0.928	0.985	0.871	0.954
LPIPS									
MobileNeRF	0.025	0.077	0.048	0.050	0.025	0.092	0.032	0.145	0.062
nvdiffrec	0.023	0.086	0.032	0.064	0.047	0.111	0.031	0.188	0.073
TensoRF (DT)	0.076	0.130	0.070	0.113	0.090	0.146	0.070	0.230	0.115
NeuS (DT)	0.033	0.101	0.065	0.041	0.056	0.084	0.021	0.191	0.074
nerf2mesh	0.046	0.084	0.045	0.060	0.047	0.107	0.042	0.145	0.072
nvdiffrec (m)	0.020	0.104	0.057	0.068	0.059	0.116	0.028	0.220	0.084
Ours (F)	0.036	0.073	0.035	0.041	0.027	0.089	0.024	0.167	0.061
Ours (HQ)	**0.026**	**0.068**	**0.033**	**0.035**	**0.023**	**0.085**	**0.017**	**0.159**	**0.056**
Ours (F-m)	0.037	0.079	0.040	0.043	0.031	0.091	0.024	0.174	0.065
Ours (HQ-m)	0.027	0.074	0.038	0.036	0.027	0.086	**0.017**	0.164	0.059

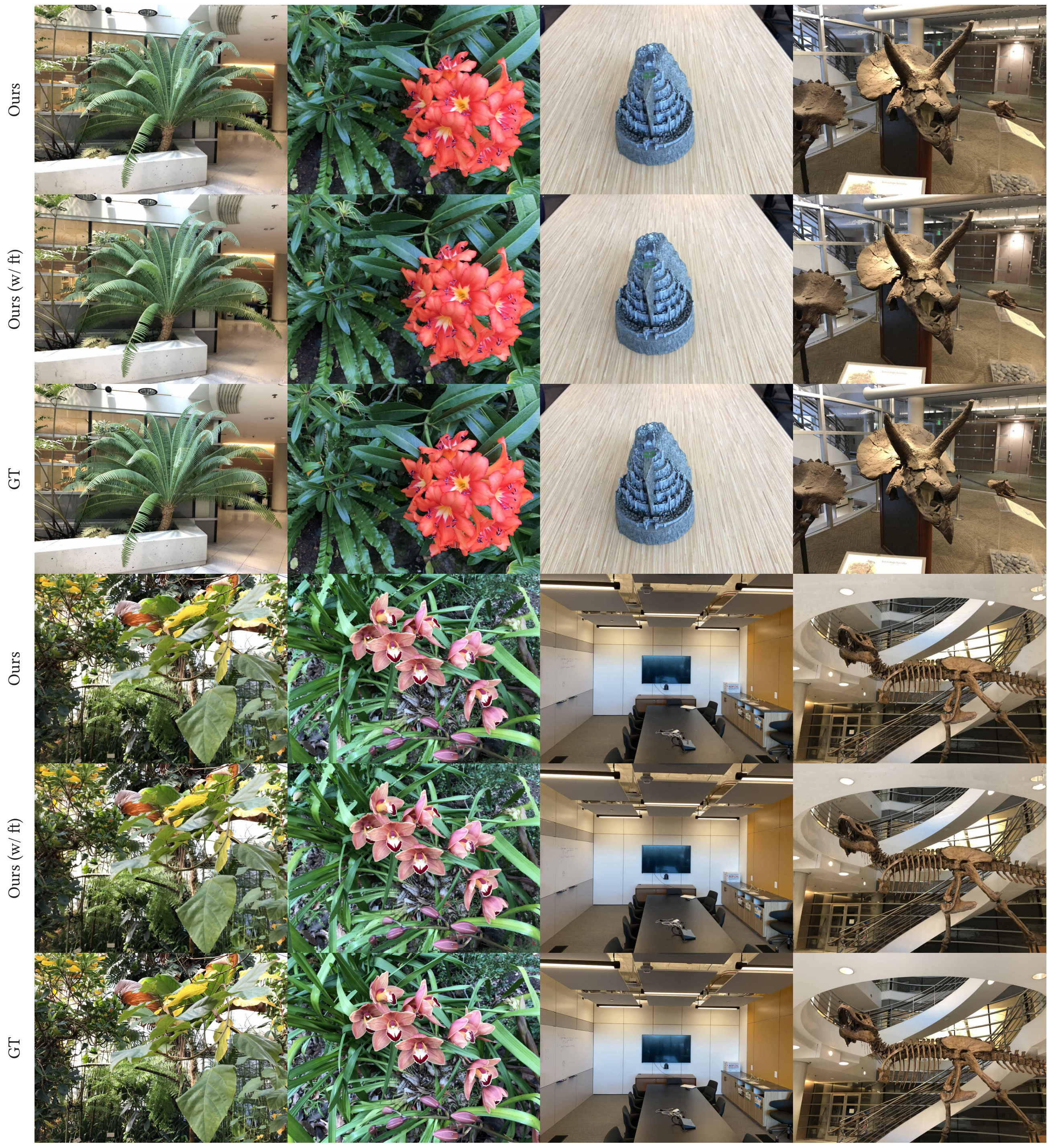

Figure 9. LLFF renderings.

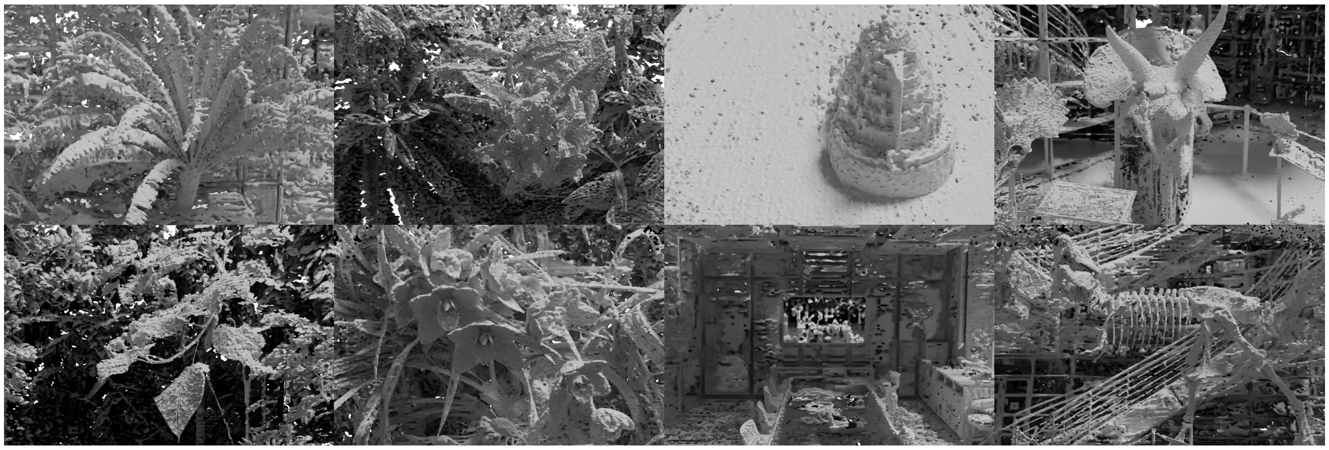

Figure 10. LLFF mesh.

Table 5. Appearance network architecture of Ours (HQ) and Ours (F) for NeRF-Synthetic.

Name	High-Quality	Fast
app matrix xy	Param (48 x 300 x 300)	Param (48 x 300 x 300)
app matrix yz	Param (48 x 300 x 300)	Param (48 x 300 x 300)
app matrix zx	Param (48 x 300 x 300)	Param (48 x 300 x 300)
app vector x	Param (48 x 300 x 1)	Param (48 x 300 x 1)
app vector y	Param (48 x 300 x 1)	Param (48 x 300 x 1)
app vector z	Param (48 x 300 x 1)	Param (48 x 300 x 1)
basis mat	Linear (144, 12, bias=False)	Linear (144, 27, bias=False)
last_layer	Linear (99, 64, bias=True) ReLU (inlace=True) Linear (64, 64, bias=True) ReLU (inlace=True) Linear (64, 3, bias=True)	Spherical Harmonics

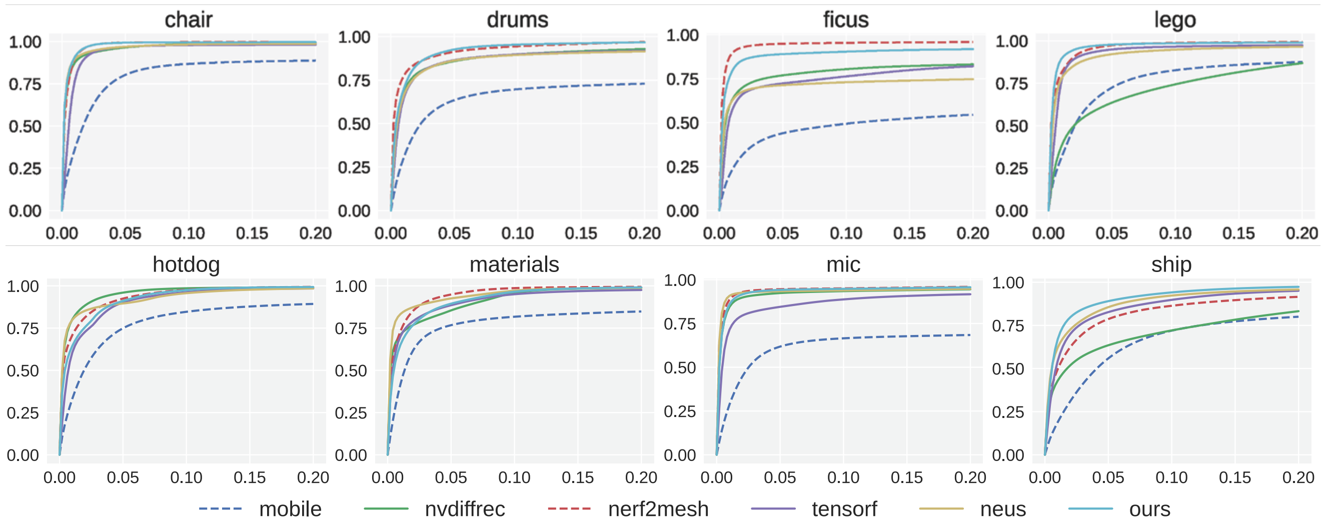

Figure 11. VSA plots for different misalignment tolerances.

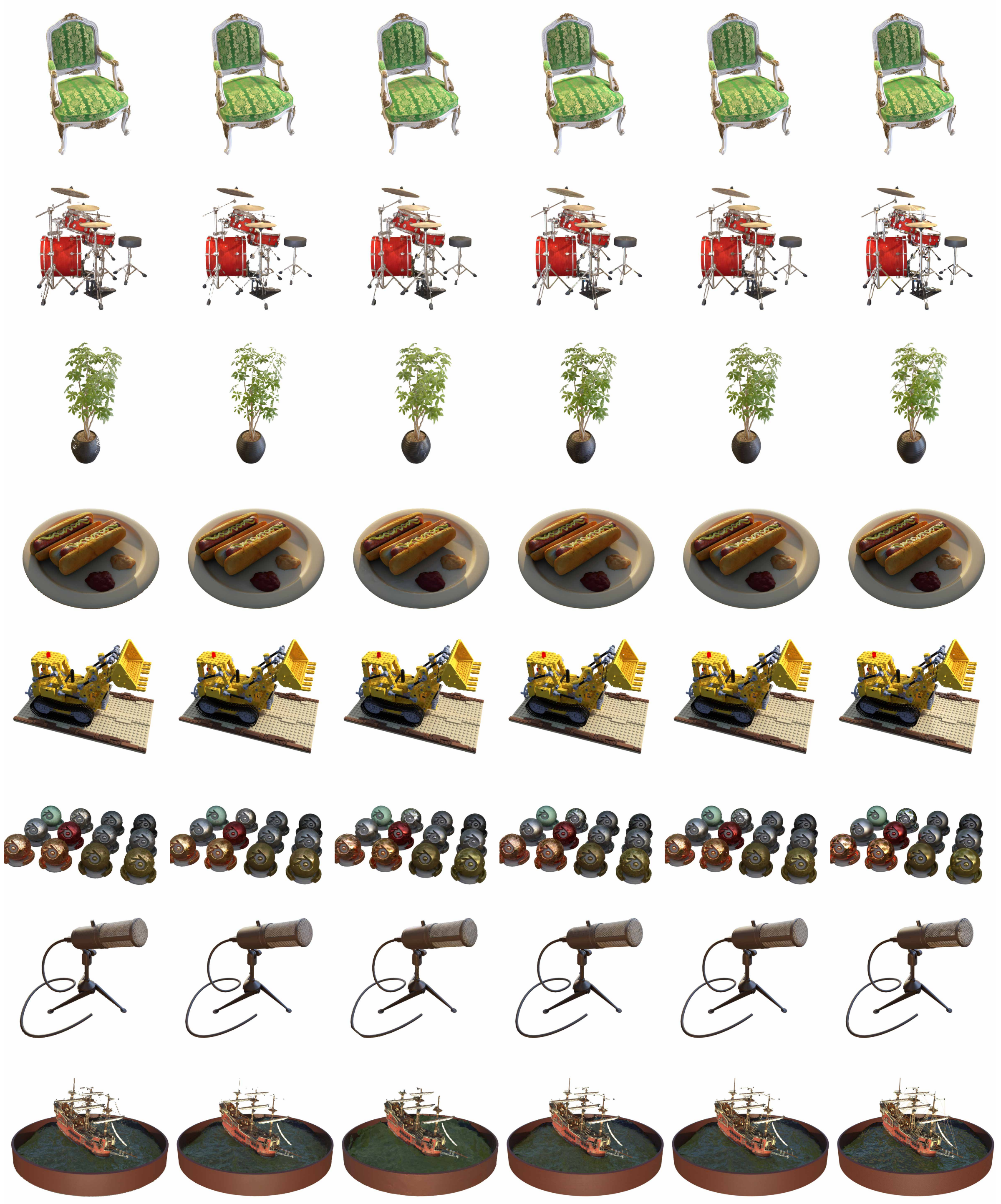

Figure 12. NeRF-Synthetic renderings.

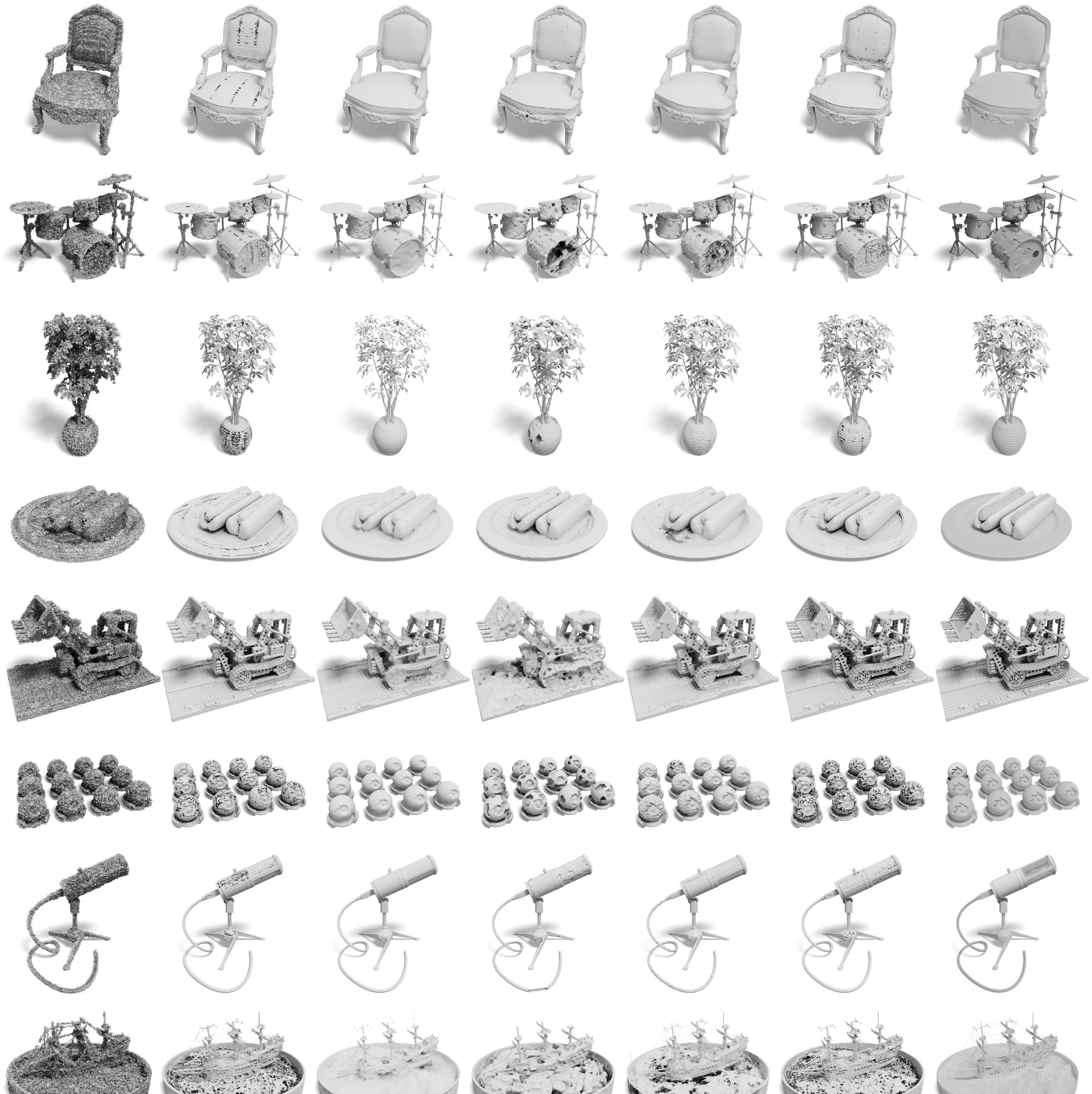

Figure 13. NeRF-Synthetic mesh.

Table 6. Trade-off between rendering speed and quality with different appearance network capacity. 8× MS: 8× sample per-pixel MSAA, 16× SS: 16× sample per-pixel SSAA.

Params	AA	PSNR↑	SSIM↑	LPIPS↓	FPS
#feat=48	8× MS	30.34	0.949	0.062	93
mlp=3×64	16× SS	31.16	0.954	0.057	26
#feat=48	8× MS	29.73	0.942	0.071	322
mlp=3×16	16× SS	30.49	0.947	0.064	86
#feat=12	8× MS	30.11	0.946	0.066	98
mlp=3×64	16× SS	30.90	0.951	0.060	27
#feat=12	8× MS	29.55	0.941	0.073	585
mlp=3×16	16× SS	30.28	0.946	0.066	163
#feat=48	8× MS	29.73	0.943	0.068	312
SH	16× SS	30.44	0.949	0.063	82

Table 7. Ablation study for Stage 2. Except for the first row using GT mesh, the rest experiments are conducted on fixed meshes extracted from pre-trained TensoRF by Marching Cubes. MLP: vanilla NeRF [6] representation; Hash: HashGrid used in iNGP [7]; SH: Spherical Harmonics; TF: TensoRF-VM.

Geo. + App.	PSNR↑	SSIM↑	LPIPS↓
GT + TF	31.78	0.958	0.053
TFmesh + MLP	26.28	0.915	0.203
TFmesh + Hash	26.62	0.921	0.090
TFmesh + SH	26.48	0.909	0.103
TFmesh + TF	27.00	0.929	0.081