Supplementary Materials for Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement

Jiaxiang Tang[1], Hang Zhou[2], Xiaokang Chen[1], Tianshu Hu[2], Errui Ding[2], Jingdong Wang[2], Gang Zeng[1]
[1]Key Lab. of Machine Perception (MoE), School of IST, Peking University. [2]Baidu Inc.
{tjx, pkucxk, zeng}@pku.edu.cn {zhouhang09,hutianshu01,dingerrui,wangjingdong}@baidu.com

A. Additional Implementation Details

Network Architecture. We use the multi-resolution hash-grid encoder [4] and shallow MLPs to cosntruct the first stage's NeRF network. The density grid E^{geo} use 16 resolution levels with each level containing 1-channel features, and a 2-layer MLP with 32 hidden channels is used to convert the features into density. The color grid E^{app} use 16 resolution levels with each level containing 2-channel features. A 3-layer MLP with 64 hidden channels convert the color features into 3-channel diffuse color and 3-channel specular features. The specular features along with view directions are fed into a 2-layer MLP with 32 hidden channels to produce the view-dependent 3-channel specular color.

Visibility culling & Mesh cleaning. In the first stage of our approach, we adopt the Marching Cubes algorithm to extract a coarse mesh from NeRF's density field. To reduce the size of the resultant mesh, we incorporate a visibility culling mechanism to eliminate vertices and faces that are invisible from all training cameras. More specifically, we cast rays from each training camera and calculate their intersection with the surface. In doing so, we trace the corresponding face and label it as visible. However, in situations where the training cameras are sparsely located, this approach may result in excessive culling. To address this issue, we apply dilation to the visible faces using a predetermined kernel size. For the NeRF-synthetic dataset, we utilize a kernel size of 5, while for the LLFF and Mip-NeRF 360 dataset, we increase it to a larger value of 50, given that training cameras may be sparse for the far background. The mesh can be further post-processed to remove floaters based on the diameter and number of faces for each connected component. We also clean the mesh by merging close vertices, removing duplicated faces, and repairing non-manifold vertices and faces [2]. In essence, these methods help to remove unnecessary vertices and faces to maintain a reasonably small mesh size.

Baking. After completing the two-stage training, we convert the appearance network into texture images for real-time rendering. Initially, the resolution of the texture image is set to 4096 for the center mesh in $[-1, 1]^3$. Subsequently, for meshes of outdoor regions, the texture resolution is decreased by a power of 2, with a minimum resolution of 1024. To eliminate seam-like texture artifacts caused by UV unwrapping [5], we repair the border of each connected component by out-painting 1 pixel on the texture image. The floating-point diffuse color and specular features in $[0, 1]$ range are quantized into 8-bit precision PNG images. Following MobileNeRF [1], we found that the rendering quality is not significantly affected through baking.

Hyper-parameters. Since different types of dataset (*e.g.*, from objects without background to unbounded scenes) can require very different hyper-parameters to maximize performance [3, 6], we explore different set of hyper-parameters, especially for loss weights. By default, we set the weight of $\mathcal{L}_{\text{TV}}$ to 1×10^{-8}, the weight of $\mathcal{L}_{\text{smooth}}$ to 1×10^{-3}, and the weight of $\mathcal{L}_{\text{offset}}$ to 0.1. The other loss weights are default to 0 unless specified. For the NeRF-synthetic dataset and the LLFF dataset, we use all the default weights. For the Mip-NeRF 360 dataset, we set the weight of $\mathcal{L}_{\text{entropy}}$ to 1×10^{-3}. The training steps for stage 1 is also set differently. We train $30,000$ steps for the NeRF-synthetic dataset, but we found $10,000$ steps are enough for the LLFF and Mip-NeRF 360 datasets to converge. For the iterative mesh refining algorithm, we apply the subdivision and decimation at $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.7\}$ ratio of total training steps. The minimum edge length for subdivision is set to 1% of the diagonal of the bounding box of the mesh (which eqauls to $0.02\sqrt{3}$ in our case). We decimate 10% of the faces with an error above e_{decimate}, and remesh them with an average edge length of 2% of the diagonal of the bounding box of the mesh ($0.04\sqrt{3}$).

B. Additional Experimental Results

B.1. Additional Qualitative Results

Relighting. Although the lighting is baked into textures in our methods, we aim to showcase that our mesh is proficient enough to execute relighting for a scene captured in predominantly ambient lighting conditions. Figure 1 exhibits

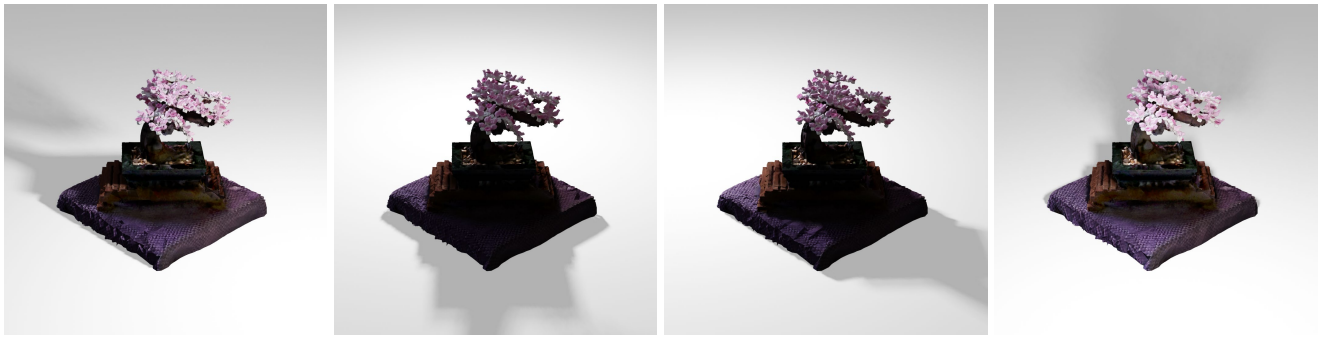

Figure 1: **Relighting**. We relight the mesh with only the diffuse texture with a point light source.

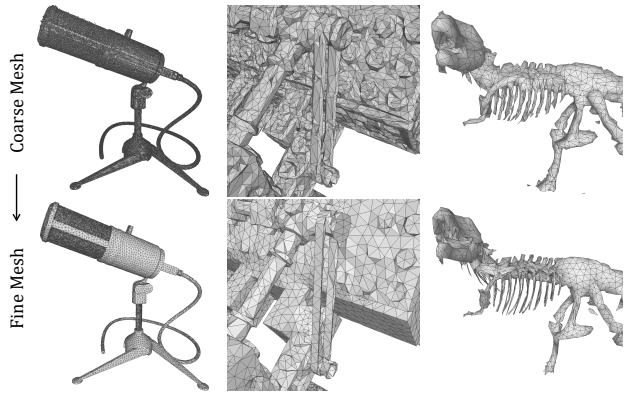

Figure 2: **Adaptive Face Density**. Our iterative mesh refining allows adaptive face density learned from data. It enhances the surface quality and reduces face counts.

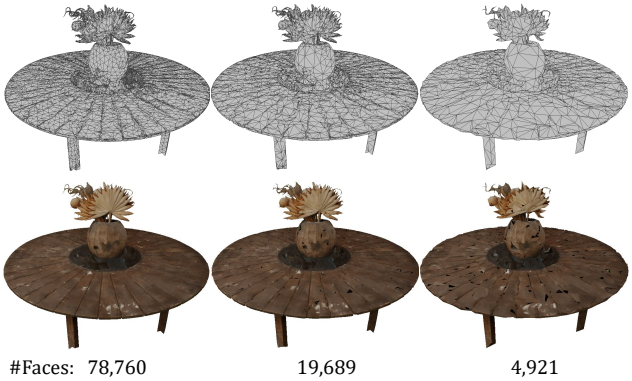

Figure 3: **Levels of details (LOD) simulation**. We decimate the reconstructed mesh to create different LODs.

a reconstructed mesh which has been relit with a rotating point light source. Only the diffuse texture is utilized.

Levels of detail (LODs) simulation. Our textured surface mesh is demonstrated to be suitable for supporting LODs in Figure 3. This can solely be accomplished with an accurate surface mesh, as decimation is primarily designed for minimizing geometric error.

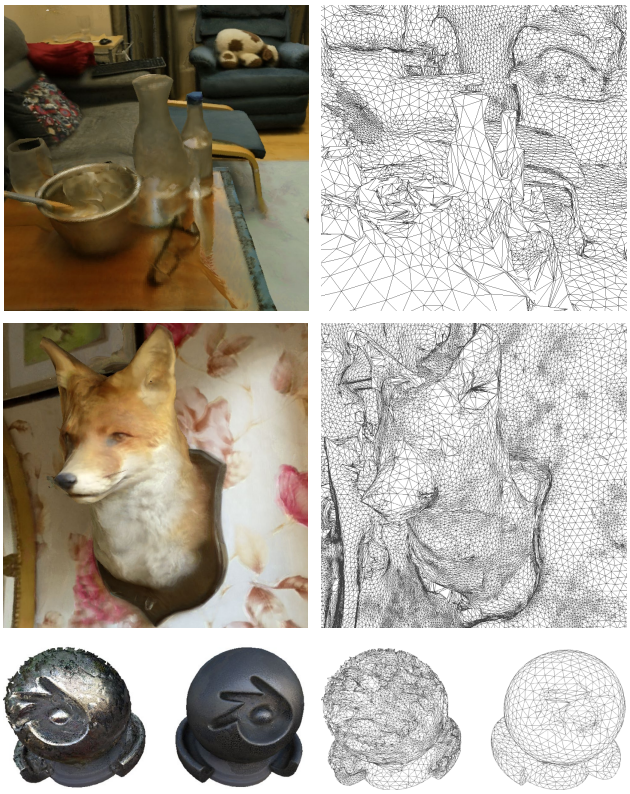

Figure 4: **Limitations**. We visualize some examples about the limitation of our method.

Additional Visualizations. We also provide more visualization on scenes with background in Figure 6. In Figure 2, we show more visualizations on the iterative mesh refining. In Figure 5, we visualize the effect of the TV loss on mesh quality.

Limitations. Our method's limitations are illustrated in Figure 4. As we solely perform single-layer rasterization, our approach is incapable of handling semi-transparent ob-

Figure 5: Ablation on TV loss. We remove some surface to show internal geometry.

jects such as glass bottles, and tends to learn an opaque texture. Animal fur, which usually requires volumetric representation for better simulation, is difficult to emulate due to the smoothness regularization of the mesh surface. Lastly, since the appearance network is relatively small, it cannot model intricate view-dependent effects. Therefore, our model tends to manipulate vertices to simulate the effects, which results in a lack of smoothness and inaccurate geometry. We are hopeful for improved decomposition of surface and materials to overcome this issue.

B.2. Additional Quantitative Results

The per-scene rendering quality evaluation results are listed in Table 2, Table 3, and Table 4. In Table 1, we perform more ablation on the regularization losses.

References

[1] 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. 1

[2] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 1

[3] 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

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 1

[5] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022. 1

[6] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *CVPR*, 2022. 1

Figure 6: **More Visualizations**. We visualize the mesh and diffuse color of more scenes from the Mip-NeRF 360 and LLFF datasets.

	PSNR↑	SSIM↑	LPIPS↓
Ours	22.36	0.493	0.478
Ours w/o $\mathcal{L}_{\text{entropy}}$	22.32	0.492	0.481
Ours w/o $\mathcal{L}_{\text{TV}}$	22.22	0.486	0.483

Table 1: We ablate the regularizations on the Mip-NeRF 360 dataset.

	Metric	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours (volume)		33.87	25.20	30.24	35.09	33.66	27.70	32.65	28.59	30.88
Ours (mesh)	PSNR↑	31.93	24.80	29.81	34.11	32.07	25.45	31.25	28.69	29.76
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		34.25	25.04	30.08	35.70	34.90	26.26	32.63	29.47	31.04
Ours (volume)		0.977	0.929	0.970	0.974	0.972	0.930	0.981	0.872	0.951
Ours (mesh)	SSIM↑	0.964	0.927	0.967	0.970	0.957	0.896	0.974	0.865	0.940
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.978	0.926	0.967	0.974	0.977	0.906	0.979	0.875	0.948
Ours (volume)		0.049	0.108	0.064	0.052	0.055	0.091	0.047	0.163	0.079
Ours (mesh)	LPIPS↓	0.046	0.084	0.045	0.060	0.047	0.107	0.042	0.145	0.072
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.031	0.084	0.046	0.058	0.025	0.111	0.038	0.138	0.066

Table 2: Rendering quality on the NeRF-synthetic dataset.

	Metric	Room	Fern	Leaves	Fortress	Orchids	Flower	Trex	Horns	Mean
Ours (volume)		31.12	25.47	20.58	30.45	20.54	27.19	28.15	27.83	26.42
Ours (mesh)	PSNR↑	29.24	23.94	19.22	28.02	19.08	26.48	25.80	26.25	24.75
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		30.03	23.21	18.71	28.96	19.34	26.10	26.41	26.40	24.90
Ours (volume)		0.939	0.802	0.700	0.887	0.668	0.823	0.910	0.866	0.824
Ours (mesh)	SSIM↑	0.914	0.751	0.644	0.765	0.602	0.879	0.868	0.819	0.780
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.923	0.709	0.621	0.859	0.607	0.797	0.879	0.831	0.778
Ours (volume)		0.201	0.248	0.253	0.167	0.270	0.204	0.180	0.223	0.218
Ours (mesh)	LPIPS↓	0.246	0.303	0.321	0.270	0.314	0.204	0.215	0.260	0.267
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.254	0.342	0.358	0.203	0.312	0.224	0.214	0.259	0.271

Table 3: Rendering quality on the LLFF dataset.

	Metric	Bicycle	Garden	Stump	Mean
Ours (volume)		20.88	23.41	22.70	22.33
Ours (mesh)	PSNR↑	22.16	22.39	22.53	22.36
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		22.28	22.86	23.08	22.74
Ours (volume)		0.469	0.567	0.578	0.538
Ours (mesh)	SSIM↑	0.470	0.500	0.508	0.493
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.479	0.551	0.540	0.523
Ours (volume)		0.545	0.419	0.478	0.481
Ours (mesh)	LPIPS↓	0.510	0.434	0.490	0.478
Ours (mesh w/o $\mathcal{L}_{\text{smooth}}$)		0.509	0.402	0.459	0.457

Table 4: Rendering quality on the Mip-NeRF 360 dataset.