LTM: Lightweight Textured Mesh Extraction and Refinement of Large Unbounded Scenes for Efficient Storage and Real-time Rendering Supplementary Material

Jaehoon Choi [1] Rajvi Shah [2] Qinbo Li [2] Yipeng Wang [2] Ayush Saraf [2] Changil Kim [2]
Jia-Bin Huang [1,2] Dinesh Manocha [1] Suhib Alsisan [2] Johannes Kopf [2]
[1]University of Maryland [2]Meta

1. Implementation Details

For geometry reconstruction in Section 3.1, the model structure of geometry model f_{sdf} consists of the grid feature and shallow MLPs. Following the BakedSDF paper, color model g_{color} adopts the appearance model of Ref-NeRF [12]. For mip-NeRF 360 dataset and Tanks & Temples, we set 0.01 to a weight for eikonal regularizaiton except for room scene. We use 0.1 for eikonal regularization for the room scene and two scenes from Deep Blending dataset [3]. For modeling appearance, we build on training techniques from mip-NeRF 360 [1] except for integrated positional encoding. We set the distortion loss weight to 0.002. Our learning schedule is 20k iterations of optimization using Adam [6], a learning rate is annealed from 1e-3 to 1e-4 To design the model structure, we adopted the appearance and density model of NeRF2Mesh [11] because this model has a lightweight structure and can generate diffuse texture and specular features. We use multiresolution hash encoding [8] with hash table size 2^{16}, coarsest resolution of 16, highest resolution of 4096, 16 levels and a number of feature dimension per entry of 2. To effectively train the specular model, we apply L1 regularization on the specular color with a weight set to 1e-5. In Section 3.2, for extracting the clean mesh, we employ the visibility and free-space culling techniques introduced by BakedSDF [14]. After Section 3.3, to export the texture, we first apply xatlas [15] to obtain per-vertex UV mapping. Subsequently, we can calculate diffuse color and specular features using f_{color} after the procedures outlined in Section 3.3. All these color maps and features are 3-channel .png files, which are lightweight compared to the large feature maps used in neural texture. For real-time rendering, we follow the same method presented by previous research [2, 4, 11].The MLP for converting specular features to color is stored in a .json file. In rendering, a fragment shader incorporates this file to apply view-dependent effects. For experiments on the mip-NeRF 360 dataset [1], we use three outdoor scenes—bicycle, garden, and stump—and four indoor scenes—room, counter, kitchen, and bonsai. These scenes include highly reflective surfaces and thin structures, posing challenges for mesh-based rendering. Additionally, we include playroom and drjohnson from the deep blending dataset [3], as well as barn and courthouse from the Tanks & Temples dataset [7]. We implement all neural implicit representations, including Geometry Reconstruction, Appearance Modeling, and Optimization components, using PyTorch [9].The mesh decimation part is implemented in C++. Furthermore, all experiments are conducted on a V100 GPU.

2. Additional Explanations

To initialize the appearance in Section 3.1, g_{color} incorporates not only the position and viewing direction but also utilizes the predicted surface normal and a 256-dimensional geometric feature as input. This 256-dimensional geometric feature and surface normal are produced by the geometry MLP f_{sdf}. When we modify the vertex position, and deformed vertices end up in empty space, the geometry MLP becomes incapable of computing a useful geometric feature, thereby impeding the rendering of accurate colors. Embedding the neural feature for each vertex is an option [13]; however, it requires significant GPU memory, particularly for large unbounded scenes. Thus, we only use the geometry MLP for extracting mesh.

For mesh decimation in Section 3.2, we group the vertices into 8 ($2 \times 2 \times 2$) clusters and calculate the quadric error for each cluster. Subsequently, we employ geometry-aware QSlim for each cluster to perform edge decimation. If the quadric error for each cluster exceeds a certain threshold, we halt the execution of QSlim.

3. Additional Qualitative Comparisons

In the manuscript, we present mesh reconstruction results exclusively for Bicycle, Garden, Counter, and Room (refer to Fig. 4 in the manuscript) due to the limited space. We show results for the remaining scenes: Stump, Kitchen, and Bonsai. We show both the mesh reconstruction and rendering results for these scenes, as shown in Fig. 1 and Fig. 2. Furthermore, we visualize both the mesh reconstruction and rendering results for Playroom and Dr. Johnson from the Tanks & Temples dataset, and for Barn and Courthouse from the Deep Blending dataset, as illustrated in Fig. 3 and Fig. 4.

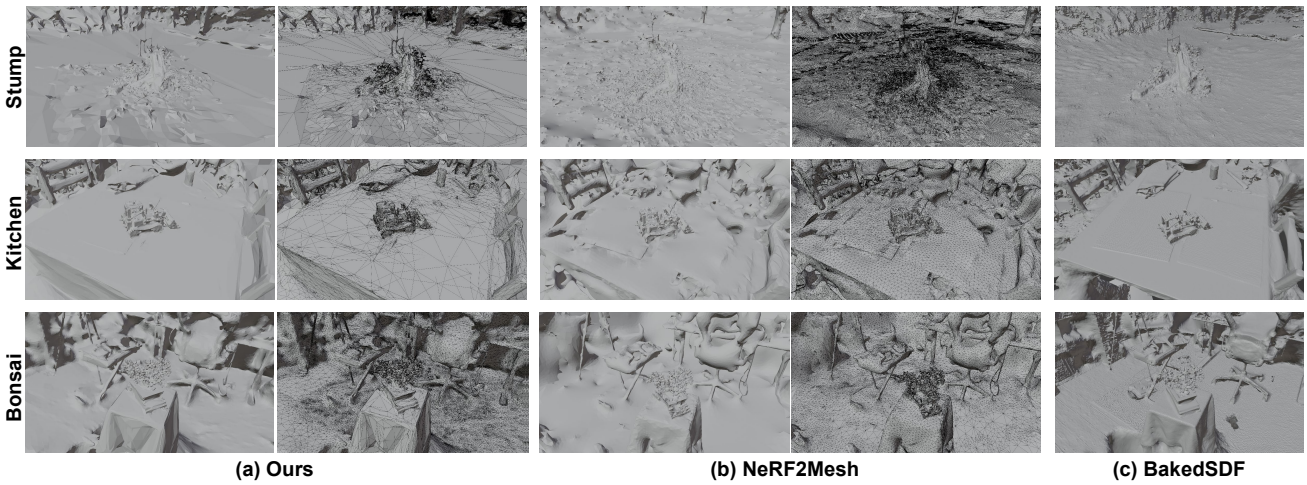

Figure 1. Qualitative comparison between (a) Ours, (b) NeRF2Mesh [11], and (c) BakedSDF [14]. Our method and NeRF2Mesh show shading mesh and wireframe mesh extracted without texture, respectively.

Figure 2. Qualitative comparison between (a) Ours, (b) NeRF2Mesh [11], (c) BakedSDF* [14], and (d) MobileNeRF [2]. The visual results of BakedSDF* generated in its first stage through volumetric rendering.

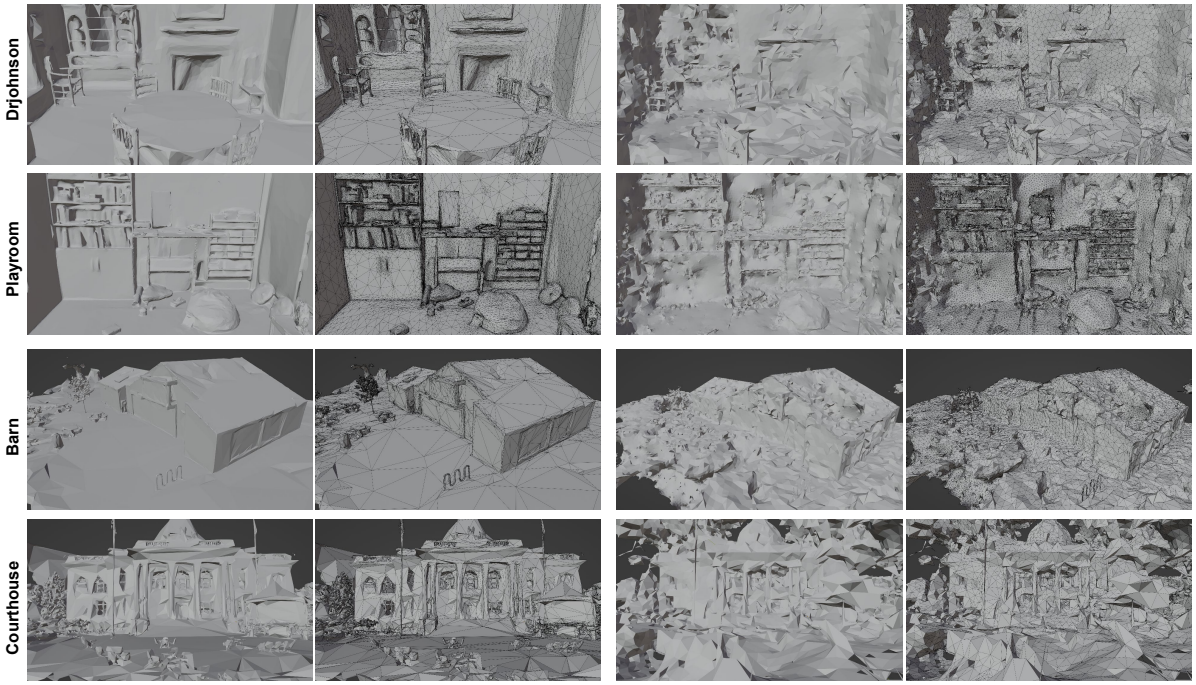

Figure 3. Qualitative comparison between (a) Ours and (b) NeRF2Mesh [11]. Our method and NeRF2Mesh show shading mesh and wireframe mesh extracted without texture, respectively.

Figure 4. Qualitative comparison between (a) Ours, (b) NeRF2Mesh [11], and (c) MobileNeRF [2].

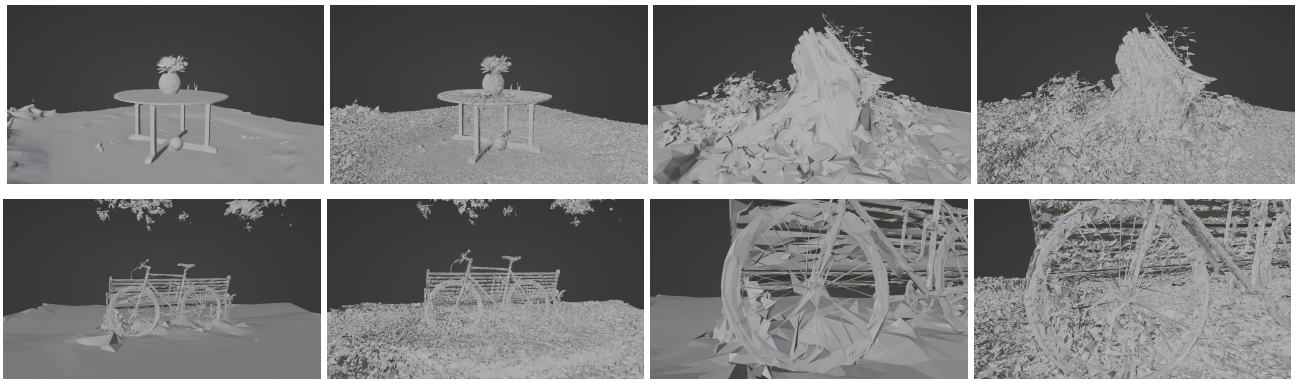

Figure 5. Comparison of mesh quality when optimized with decimation (columns 1, 3) vs. without decimation (columns 2, 4).

PSNR	Garden	Bicycle	Stump
with Decimation	23.5	22.4	23.7
w/o Decimation	24.1	22.8	24.2

(a) Decimation Ablation

CD ↓	Ours	NeRF2Mesh	BakedSDF*
garden	0.01 / 0.13	1.04 / 0.86	- / 0.12
bicycle	0.02 / 0.15	1.28 / 0.90	- / 0.11

(b) Geometry Evaluation

Table 1. (a) Rendering quality improves w/o decimation, but the resulting mesh suffers from distortions (see Fig1) (b) Chamfer distance (CD) with respect to two baselines - BakedSDF* (from SDFStudio) / MVS. Our scores are significantly better than NeRF2Mesh and only slightly worse than BakedSDF*.

4. Role of Decimation and Ablation

We demonstrate the effectiveness of mesh decimation within our pipeline. In Fig. 5, We visualize the mesh results obtained using our method both with and without mesh decimation. Except for aggressive mesh decimation, we adhere to the same pipeline as our method (without decimation). We utilize a naive QSlim approach, employing a 50 percent mesh decimation, as training large mesh vertices is restricted by GPU memory during differentiable rendering. Training vertex deformation with a large number of vertices shows more geometric details (bicycle wheel and stump trunk in Fig. 5). This detailed geometry enhances rendering quality. However, vertex deformation can result in bumpy and noisy surfaces, particularly in smooth regions (as depicted in the ground regions in Fig. 5). We provide the ablation comparing the visual quality with and without the decimation step in Table1-(a). In this Table, optimization with a larger number of vertices demonstrates superior rendering quality due to its more detailed geometry. However, it requires significant disk storage and presents noisy surfaces across most regions.

Since the Mip-NeRF 360 dataset does not provide the ground truth meshes, conducting quantitative geometric evaluations proves challenging. Hence, we reconstruct the two baseline meshes using two different methods. First, BakedSDF* can generate smooth and highly detailed geometry with a large number of vertices, which we consider as the ground truth mesh. Second, we employ multiview stereo [10] and eliminate noisy points using a statistical point removal method. We then utilize screened Poisson surface reconstruction [5] to generate the mesh, which we regard as the ground truth mesh. In Table 1-(b), we show the Chamfer distance with respect to two baselines (BakedSDF* / MVS). Our scores outperform NeRF2Mesh significantly and are only marginally lower than those of BakedSDF*. While this evaluation may lack precision, these comparisons offer quantitative insights that align with the qualitative trends observed in Figure 4 of the manuscript.

5. Thorough Discussion of Limitations

In this section, we provide a more thorough discussion of limitations. First, we can design more sophisticated appearance model for representing diffuse color and view-dependency effects. BakeSDF adopts a spherical Gaussians to represent view-depdency effects. Our method utilizes a shallow MLP with a lightweight specular feature map and does not incorporate any embedding representation for the viewing direction. A more meticulous design of the appearance model can result in improved rendering quality. Second, training vertex deformation using the ADAM optimizer often results in mesh distortions including self-intersections and flipped triangles (as seen in Fig. 5). While rotation-invariant ADAM (RADAM mitigates mesh distortion to some extent, we still observe similar distortion in the mesh. Currently, the ADAM optimizer stands as the optimal choice available to us. In the future, it is imperative to develop careful optimizer to mitigate mesh distortion, or alternatively, necessitate the incorporation of robust regularization techniques to facilitate the generation of high-quality

manifold meshes.

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. 1

[2] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023. 1, 2, 3

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

[4] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 1

[5] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 4

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[7] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 1

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 1

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[10] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 4

[11] 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*, 2022. 1, 2, 3

[12] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 1

[13] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision*, pages 597–614. Springer, 2022. 1

[14] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv*, 2023. 1, 2

[15] Jonathan Young. xatlas. 1