# Multi-View Mesh Reconstruction with Neural Deferred Shading Supplementary Material

Markus Worchel[1,2*] Rodrigo Diaz[1,3*] Weiwen Hu[1]
Oliver Schreer[1] Ingo Feldmann[1] Peter Eisert[1,4]

[1]Fraunhofer HHI [2]TU Berlin [3]Queen Mary University of London [4]HU Berlin

## 1. Ethical Considerations

Our method has the same potential for misuse as other multi-view 3D reconstruction pipelines. For example, it could be used to digitally reconstruct dangerous objects (*e.g.* weapons, weapons parts, ammunition) for reproduction in 3D printing.

Because we also train an appearance model, a malicious actor could potentially modify or edit materials of a reconstructed scene, for example to create fakes by modifying the skin color of a person.

With publicly available code, means for mitigation would be difficult to implement from our side. However, we feel that the overall potential for misuse is low and there are few obvious paths to malicious use or potential damage. Also, our method requires a set of calibrated images, thus is much less accessible than 3D reconstruction or deep fakes based on single images.

## 2. Dataset Details

### 2.1. DTU MVS Dataset

Our evaluation is based on the scripts included in the official DTU MVS dataset release [5]. In our experiments, we use a derived dataset [23] that includes object masks and was assembled in the context of prior work [14, 25]. Our implementation expects the input in a different file structure and we will release the converted dataset with our paper.

We are not aware of any copyright or license attached to the DTU dataset. This dataset is widely used in the scientific community.

### 2.2. Human Body Dataset

The human body dataset used in the mesh refinement experiments consist of recordings of two persons, taken in a volumetric capture studio [19]. The subjects are hired actors and they consented in writing to be recorded and to their data being used for research purposes. The consent covers showing the subjects' faces in publications.

*Equal contribution

## 3. Implementation Details

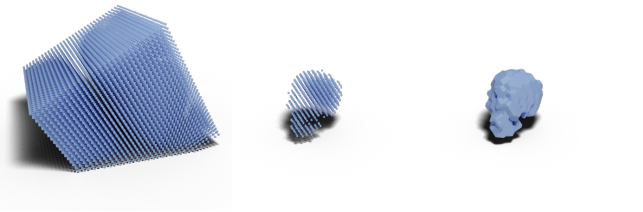



Figure 1. Construction of the initial visual hull mesh for 3D reconstruction.

Our implementation is built on top of the automatic differentiation framework PyTorch [15] and includes code released in the context of previous publications [9, 13, 16, 22]. For remeshing we use a library with Python bindings [21]; for differentiable rasterization we use the high-performance primitives by Laine *et al.* [12]. Additionally, we rely on a variety of other libraries [1–4, 6–8, 10, 11].

In our implementation, we do not assume normalized camera positions. However, we require a rectangular bounding box to normalize the domain to a cube centered at $(0, 0, 0)$ with side length 2.

For 3D reconstruction, we also use the bounding box to construct the initial mesh (Figure 1): we place a grid of points ($32 \times 32 \times 32$) inside the bounding box volume and project the points into each camera image. If a point lies outside any image or mask region, it is removed. We reconstruct a mesh surface from the remaining points with marching cubes.

Table 1. Full hierarchical runtime decomposition of our method for DTU scan 122 ("Owl") with 2000 iterations. Invocations of the neural shader are included in the shading term. Some one-time operations before to the optimization loop are not explicitly listed (*e.g.* initialization of the neural network, optimizers, and the rasterizer). In this experiment, we pre-computed the initial mesh. Computing the visual hull takes roughly 0.5 seconds.

| Operation | Time [s] |
|---|---|
| Total | 341.03 |
| ↳ Reading data | 20.54 |
| ↳ Images | 20.50 |
| Mesh | 0.04 |
| Optimization loop | 312.16 |
| ↳ Applying vertex offsets | 2.26 |
| Sampling views | 0.63 |
| Creating g-buffers | 11.49 |
| ↳ Rasterization | 6.86 |
| Interpolating coverage | 1.50 |
| Interpolating positions | 1.45 |
| Interpolating normals | 1.43 |
| Computing objective function | 135.63 |
| ↳ Silhouette | 0.73 |
| Shading | 128.50 |
| Laplacian | 3.58 |
| Normal consistency | 1.37 |
| Aggregating terms | 1.24 |
| Computing gradients | 129.12 |
| Performing descent step | 5.79 |
| Remeshing | 23.28 |

## 4. Experiment Details

**Optimization.** In the 3D reconstruction experiments, we set the gradient descent step size to $10^{-3}$ for both the mesh vertices and the neural shader. For mesh refinement, we use a step size of $10^{-4}$ for the vertices and $2 \cdot 10^{-3}$ for the shader, progressing the shader faster than the mesh.

**Reconstruction Baselines.** For COLMAP [17, 18], we use the official release 3.6 with CUDA support [20]. Similar to prior work [14], we clean the dense point clouds with masks. We also perform trimming after the screened Poisson surface reconstruction ("trim7").

For IDR [25], we use the official implementation [24] and run the reconstruction experiments with camera training, using the `dtu_trained_cameras.conf` configuration.

**Runtime Decomposition.** In the main work, we compare the runtime of one gradient descent iteration of our method to the one of IDR [25]. The comparison uses data produced by an extensive profiling mode that we implemented for our method and a simpler mode implemented on top of IDR. When profiling, we disable all intermediate outputs (*e.g.* visualizations). In the case of IDR, we profile calls to `RayTracing.forward` and inside these calls those to `ImplicitNetwork.forward`, obtaining measurements for *Geometry rendering* and *SDF evaluation*, respectively.

For our method, we record the runtime of most operations during reconstruction. Table 1 shows the full decomposition for one sample from the DTU dataset. Computing the shading term of our objective function and computing the gradients with back propagation are the main bottlenecks of our method.

## 5. Additional Experimental Results

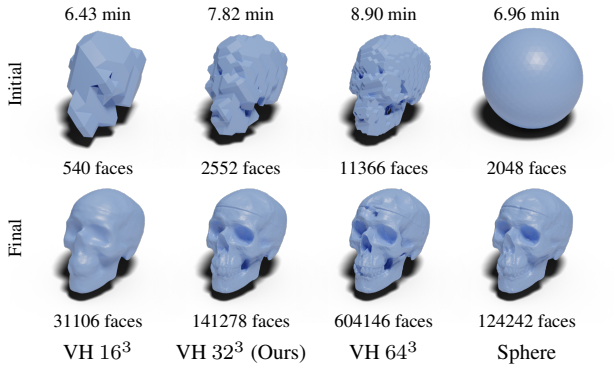



Figure 2. Reconstruction results for different initial meshes. We vary the resolution of the grid used to build the visual hull (VH) and also start from a sphere.

**Ablation Study of Initial Mesh.** We investigate how initial meshes with different resolutions, topology, and geometric distance to the target affect the reconstruction (Figure 2). Very coarse initial meshes result in missing details, while fine meshes provide too much geometric freedom, which leads to artifacts. Since we do not support topology changes, holes are only reconstructed if they are present in the initial geometry.

The initial meshes for all DTU objects consist of around 2000 triangles and the output meshes of around 100000 triangles. In terms of scalability, our method handles high-resolution meshes efficiently: in the last 500 iterations (with 2000 iterations in total), the optimization runs on a mesh with the output resolution and still performs fast gradient descent steps.

**Ablation Study of Objective Function.** We investigate the influence of the individual terms of our objective function (Figure 3). Without the Laplacian term, we observe

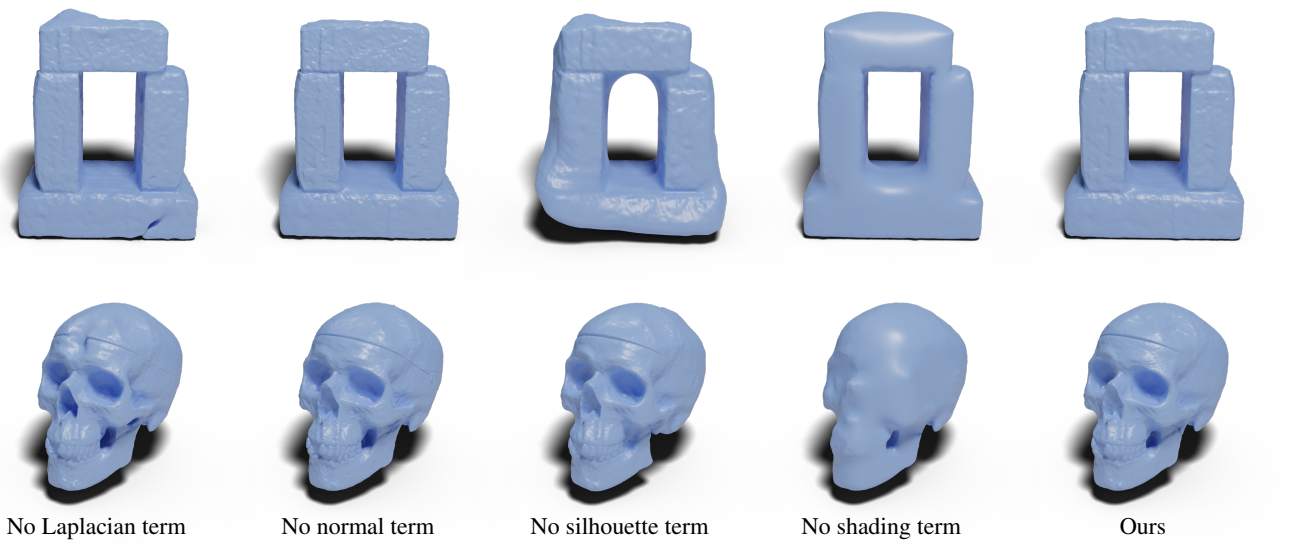



Figure 3. Ablation study of the objective function.

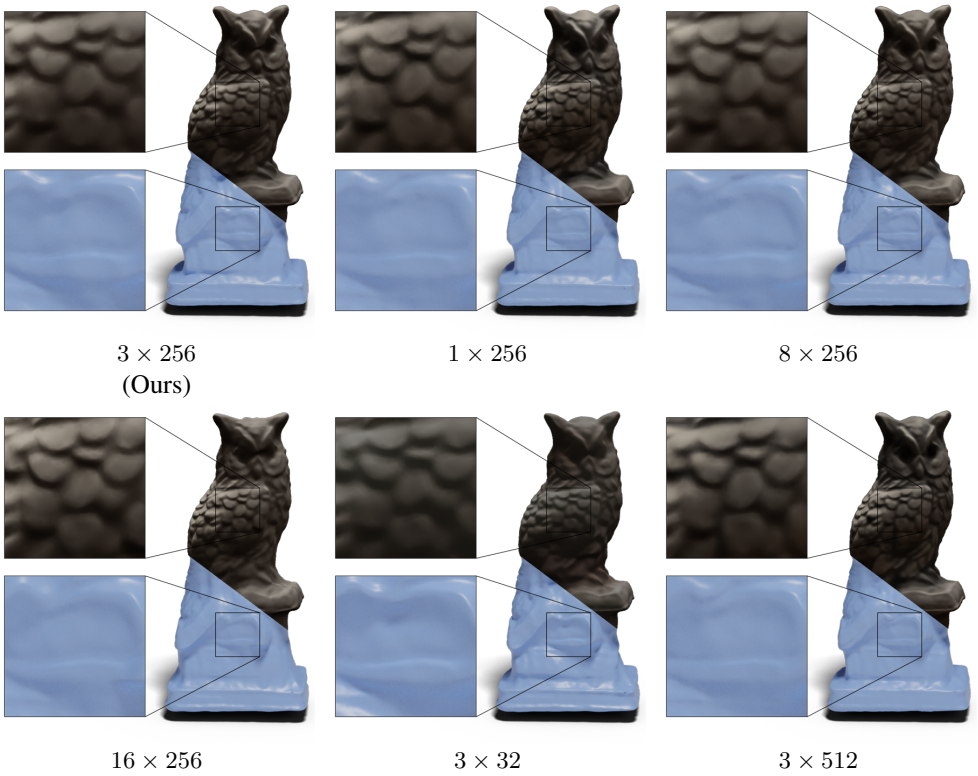
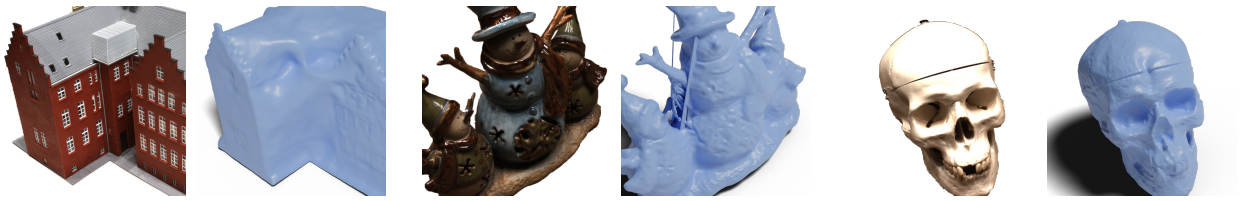



Figure 4. Different sizes of the positional part of the neural shader (number of layers × width of layers).

noticeable bumps and crack-like artifacts. The normal term has a small influence but improves smoothness around edges. Without the silhouette term, the object boundaries are not properly reconstructed. Without the shading term, the reconstructed shape resembles a smooth visual hull, missing almost all details.

**Ablation Study of Network Architecture.** In Figure 4, we show the results for different architecture configurations. Using very few units per layer leads to sharper geometry while the opposite leads to a smoother surface. In the first case, a sharper geometry does not equate with a better estimation of the surface (e.g., shadows are baked into the



Deep concavities | Failed remeshing | Overexposed regions

Figure 5. Failure cases of our reconstruction.

geometry). In the latter case, we obtain a smooth geometry but we lose geometrical sharpness. Moreover, with the increase in the network parameters, the optimization time grows accordingly.

We found that using 3 layers with 256 units per layers is a good compromise between network complexity and expressive power and yields the best results.

For SIREN [22], we noticed that the optimization procedure diverges with our default learning rate. We lowered it to $10^{-4}$ to obtain meaningful results. Despite the lower learning rate, SIREN still converges quickly.

## 6. Interactive Viewer

Since the renderer we use for optimization has the layout of a standard real-time graphics pipeline, the shader part (*i.e*. the neural shader) can be readily integrated into other graphics pipelines after training, allowing the interactive synthesis of novel views.

As an example, we implemented a "neural" viewer that uses OpenGL to rasterize positions and normals. Then, we use OpenGL-CUDA interoperability and PyTorch to shade the buffers with a pre-trained neural shader. We can envision an exciting extension where the shader is directly compiled to GPU byte code and used similar to other shaders written in high-level shading languages.

## 7. Failure Cases

Figure 5 shows some failure cases of our reconstruction method. Since the reconstruction starts from a visual hull-like mesh, deep concavities require large movements in the mesh. This movement is driven by relatively weak shading gradients, which cannot recover these concavities before the mesh becomes overly stiff (*e.g*. because the gradient descent step size is reduced during remeshing). On the other hand, silhouette gradients are larger than shading gradients, so the mesh easily "grows" to fill the masks.

We also observed that the remeshing operation sometimes produces tangled meshes for some objects. This failure case is unrecoverable and the reconstruction needs to be restarted. We are investigating the issue and will coordinate with the authors of the remeshing library.

Some regions show weak structure in the majority of images for a variety of reasons, for example because they are overexposed. Since we randomly sample one camera view per iteration, there is a high probability to select an image with low structure information for these regions. Using such a view can drive the regions' vertices in unexpected directions away from the actual surface. If such movement occurs at the wrong time (*e.g*. right before remeshing), the optimization often fails to correct those vertices.

## References


[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 1

[2] Alex Clark et al. Pillow (pil fork). https://github.com/python-pillow/Pillow, 2021. 1

[3] Michael Dawson-Haggerty et al. Trimesh. https://github.com/mikedh/trimesh. 1

[4] Szabolcs Dombi. Moderngl, high performance python bindings for opengl 3.3+. https://github.com/moderngl/moderngl. 1

[5] DTU. MVS Data Set – 2014. https://roboimagedata.compute.dtu.dk/?page_id=36. 1

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 1

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 1

[8] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library. https://libigl.github.io/, 2018. 1

[9] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, SA '20, New York, NY, USA, 2020. Association for Computing Machinery. 1

[10] Almar Klein et al. Imageio. https://github.com/imageio/imageio. 1

[11] Holger Krekel, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, Brianna Laugher, and Florian Bruhin. pytest. https://github.com/pytest-dev/pytest, 2004. 1

[12] 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

[13] 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

[14] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[16] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1

[17] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[18] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[19] Oliver Schreer, Ingo Feldmann, Sylvain Renault, Marcus Zepp, Markus Worchel, Peter Eisert, and Peter Kauff. Capture and 3d video processing of volumetric video. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4310–4314, 2019. 1

[20] Johannes Lutz Schönberger et al. COLMAP Release 3.6. https://github.com/colmap/colmap/releases/tag/3.6. 2

[21] Silvia Sellán and Baptiste Nicolet. Libigl botsch-kobbelt local remesher. https://github.com/sgsellan/botsch-kobbelt-remesher-libigl, 2021. 1

[22] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 1, 4

[23] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Derived DTU MVS Dataset. https://www.dropbox.com/sh/5tam07ai8ch90pf/AADniBT3dmAexvm_J1oL__uoa. 1

[24] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. IDR Code. https://github.com/lioryariv/idr. 2

[25] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2