Multi-View Mesh Reconstruction with Neural Deferred Shading

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

Abstract

We propose an analysis-by-synthesis method for fast multi-view 3D reconstruction of opaque objects with arbitrary materials and illumination. State-of-the-art methods use both neural surface representations and neural rendering. While flexible, neural surface representations are a significant bottleneck in optimization runtime. Instead, we represent surfaces as triangle meshes and build a differentiable rendering pipeline around triangle rasterization and neural shading. The renderer is used in a gradient descent optimization where both a triangle mesh and a neural shader are jointly optimized to reproduce the multi-view images. We evaluate our method on a public 3D reconstruction dataset and show that it can match the reconstruction accuracy of traditional baselines and neural approaches while surpassing them in optimization runtime. Additionally, we investigate the shader and find that it learns an interpretable representation of appearance, enabling applications such as 3D material editing.

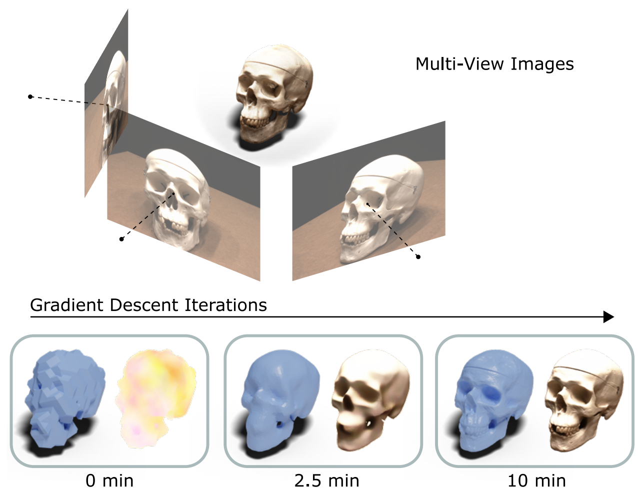

Figure 1. We reconstruct an object from images by simultaneously deforming a triangle mesh and optimizing a neural shader, comparing the renderings to the input images.

1. Introduction

The reconstruction of 3D objects based on multiple images is a long standing problem in computer vision. Traditionally, it has been approached by matching pixels between images, often based on photo-consistency constraints or learned features [16, 25]. More recently, *analysis-by-synthesis*, a technique built around the rendering operation, has re-emerged as a promising direction for reconstructing scenes with complex illumination, materials and geometry [34, 36, 37, 39, 41, 65]. At its core, parameters of a virtual scene are optimized so that its *rendered appearance* from the input camera views matches the camera images. If the reconstruction focuses on solid objects, these parameters usually include a representation of the object surface.

In gradient descent-based optimizations, analysis-by-synthesis for surfaces is approached differently depending on the differentiable rendering operation at hand. Methods that physically model light transport typically build on prior information such as light and material models [35, 36]. It is common to represent object surfaces with triangle meshes and use differentiable path tracers (*e.g.*, [29, 42, 67]) to jointly optimize the geometry and parameters like the light position or material diffuse albedo. Due to the inherent priors, these methods do not generalize to arbitrary scenes.

Other methods instead model the rendering operation with neural networks [41, 43, 64], *i.e.*, the interaction of material, geometry and light is partially or fully encoded in the network weights, without any explicit priors. Surfaces are often represented with implicit functions or more specifically *implicit neural representations* [33, 41, 44] where the indicator function is modeled by a multi-layer perceptron (MLP) or any other form of neural network and optimized with the rendering networks in an end-to-end fashion.

While fully neural approaches are general, both in terms of geometry and appearance, current methods exhibit excessive runtime, making them impractical for domains that

*Equal contribution

handle a large number of objects or multi-view video (*e.g.* of human performances [6, 15, 50, 53, 59]).

We propose Neural Deferred Shading (NDS), a fast analysis-by-synthesis method that combines triangle meshes and neural rendering. The rendering pipeline is inspired by real-time graphics and implements a technique called *deferred shading* [7]: a triangle mesh is first rasterized and the pixels are then processed by a neural shader that models the interaction of geometry, material, and light. Since the rendering pipeline, including rasterization and shading, is differentiable, we can optimize the neural shader and the surface mesh with gradient descent (Figure 1). The explicit geometry representation enables fast convergence while the neural shader maintains the generality of the modeled appearance. Since triangle meshes are ubiquitously supported, our method can also be readily integrated with existing reconstruction and graphics pipelines. Our technical contributions include:

- A fast analysis-by-synthesis pipeline based on triangle meshes and neural shading that handles arbitrary illumination and materials
- A runtime decomposition of our method and a state-of-the-art neural approach
- An analysis of the neural shader and the influence of its parameters

2. Related Work

2.1. Multi-View Mesh Reconstruction

There is a vast body of work on image-based 3D reconstruction for different geometry representations (*e.g.* voxel grids, point clouds and triangle meshes). Here, we will only focus on methods that output meshes and refer to Seitz *et al*. [51] for an overview of other approaches.

Photo-Consistency. During the past decades, multi-view methods have primarily exploited photo-consistency across images. Most of these approaches traverse different geometry representations like depth maps or point clouds before extracting (and further refining) a mesh, *e.g.* [2, 5, 13, 14, 20, 54, 58, 60]. Some methods directly estimate a mesh by deforming or carving an initial mesh (*e.g.* the visual hull) while minimizing an energy based on cross-image agreement [10–12, 21, 68]. Recently, learned image features and neural shape priors have been used to drive the mesh deformation process [30, 62]. Our method is similar to full mesh-based approaches in the sense that we do not use intermediate geometry representations. However, we also do not impose strict assumptions on object appearance across images, which enables us to handle non-Lambertian surfaces and varying light conditions.

Analysis-by-Synthesis. More than 20 years ago, Rockwood and Winget [47] proposed to deform a mesh so that *synthesized* images match the input images. Their early analysis-by-synthesis method builds on an objective function with similar terms as ours (and many modern approaches): shading, silhouette, and geometry regularization. Later works propose similar techniques (*e.g.* [8, 63, 66]), yet all either assume known material or light parameters or restrict the parameter space with prior information, *e.g.* by assuming constant material across surfaces. In contrast, we optimize all parameters of the virtual scene and do not assume specific material or light models.

Optimizing many parameters of a complex scene, including geometry, material and light, has only lately become practical, arguably with the advent of differentiable rendering. Differentiable path tracers have been used on top of triangle meshes to recover not only the geometry but also the (spatially varying) reflectance and light [35, 36], only from images. Related techniques can reconstruct transparent objects [37]. Similarly, we perform analysis-by-synthesis by optimizing a mesh with differentiable rendering. However, we use rasterization and do not simulate light transport. In our framework, the view-dependent appearance is learned by a neural shader, which neither depends on material or light models nor imposes constraints on the acquisition setup (*e.g.* co-located camera and light).

Besides mesh reconstruction from real world images, analysis-by-synthesis with differentiable rendering has recently been used for image-based geometry processing [32] and appearance-driven mesh simplification [19]. Similar to us, these approaches deform a triangle mesh to reproduce target images, albeit their targets are fully synthetic.

2.2. Neural Rendering and Reconstruction

In this work, we understand neural rendering as training and using a neural network to synthesize color images from 2D input (*e.g.* semantic labels or UV coordinates), recently named "2D Neural Rendering" [56]. Neural rendering has been used as integral part of 3D reconstruction methods with neural scene representations.

Introduced by Mildenhall *et al*. [39], neural radiance fields are volumetric scene representations used for 3D reconstruction, which are trained to output RGB values and volume densities at points along rays casted from different views. This idea has been adapted by a large number of recent works [9]. Although not strictly based on neural rendering, these methods are related to ours by their analysis-by-synthesis characteristics. While the volumetric representation can handle transparent objects, most methods focus on view synthesis, so extracted surfaces lack geometric accuracy. Lassner *et al*. [27] propose a volumetric representation based on translucent spheres, which are shaded with neural rendering. Similar to us, they jointly optimize geom-

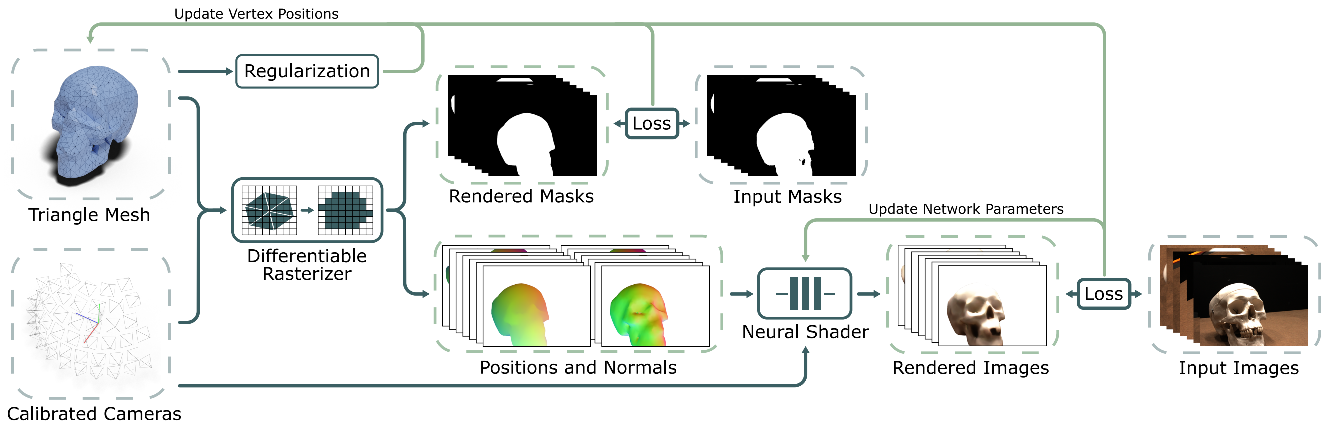

Figure 2. Overview of our optimization procedure. We rasterize the triangle mesh and shade the results with a neural network to synthesize an image for each input camera view. The shader is updated based on the difference between rendered and input images, whereas the mesh vertices are also updated based on the silhouette and a geometric regularization term. We optimize using gradient descent.

etry and appearance with a focus on speed, yet most details in their reconstruction are not present in the geometry but "hallucinated" by the neural renderer.

Implicit surfaces encoded in neural networks are another popular geometry representation for 3D reconstruction, most notably occupancy networks [38, 41, 43, 46] and neural signed distance functions [23, 44, 61, 64]. Here, surfaces are implicitly defined by level sets. For 3D reconstruction, these geometry networks are commonly trained end-to-end with a neural renderer to synthesize scenes that reproduce the input images. We also use neural rendering to model the appearance, but represent geometry explicitly with triangle meshes, which can be efficiently optimized and readily integrated into existing graphics workflows.

Similar to us, Thies *et al.* [57] present a deferred mesh renderer with neural shading. However, their convolutional neural network-based renderer can "hallucinate" colors at image regions that are not covered by geometry, as opposed to our MLP-based shader. Most notably, their method aims at view synthesis, therefore only the renderer weights are optimized, while the mesh vertices remain unchanged.

3. Method

Given a set of images $\mathcal{I} = \{I_1, \cdots, I_n\}$ from calibrated cameras and corresponding masks $\mathcal{M} = \{M_1, \cdots, M_n\}$, we want to estimate the 3D surface of an object shown in the images. To this end, we follow an analysis-by-synthesis approach: we find a surface that reproduces the images when rendered from the camera views. In this work, the surface is represented by a triangle mesh $\mathcal{G} = (V, \mathcal{E}, \mathcal{F})$, consisting of vertex positions V, a set of edges $\mathcal{E}$, and a set of faces $\mathcal{F}$. We solve the optimization problem using gradient descent and gradually deform a mesh based on an objective function that compares renderings of the mesh to the input images.

Faithfully reproducing the images via rendering requires an estimate of the surface material and illumination if we simulate light transport, *e.g.* with a differentiable path tracer [29, 42]. However, because our focus is mainly on the geometry, we do not accurately estimate these quantities and thus also avoid the limitations imposed by material and light models. Instead, we propose a differentiable mesh renderer that implements a deferred shading pipeline and handles arbitrary materials and light settings. At its core, a differentiable rasterizer produces geometry maps per view, which are then processed by a learned shader. See Figure 2 for an overview.

3.1. Neural Deferred Shading

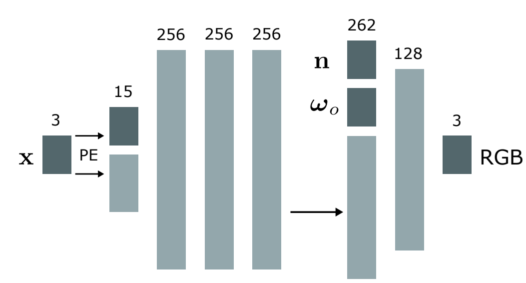

Figure 3. Architecture of the neural shader. The position $\mathbf{x}$ is transformed by a positional encoding (PE) [55] and processed by 3 fully-connected layers. The resulting feature vector is concatenated with the surface normal $\mathbf{n}$ and the view direction $\boldsymbol{\omega}_o$ and processed by the last two layers, yielding a color value. We use ReLU activations for the hidden layers and a sigmoid activation for the last layer.

Our differentiable mesh renderer follows the structure of a deferred shading pipeline from real-time graphics: Given a camera i, the mesh is rasterized in a first pass, yielding a triangle index and barycentric coordinates *per pixel*. This information is used to interpolate both vertex positions and vertex normals, creating a geometry buffer (g-buffer) with per-pixel positions and normals. In a second pass, the g-buffer is processed by a learned shader

$$f_\theta(\mathbf{x}, \mathbf{n}, \boldsymbol{\omega}_o) \in [0,1]^3 \tag{1}$$

with parameters θ. The shader returns an RGB color value for a given position $\mathbf{x} \in \mathbb{R}^3$, normal $\mathbf{n} \in \mathbb{R}^3$, and view direction $\boldsymbol{\omega}_o = \frac{\mathbf{c}_i - \mathbf{x}}{\|\mathbf{c}_i - \mathbf{x}\|}$, with $\mathbf{c}_i \in \mathbb{R}^3$ the center of camera i. It encapsulates the appearance, *i.e.*, interaction of geometry, material and light as well as the camera pixel response, and is optimized together with the geometry. We represent the shader as a shallow multi-layer perceptron, with θ as the parameters of the fully-connected layers (Figure 3). In this context, it has been shown that providing the normal and view direction with the position is necessary for disentangling the geometry from the appearance [64].

In addition to a color image, the renderer also produces a mask that indicates if a pixel is covered by the mesh.

3.2. Objective Function

Finding an estimate of shape and appearance formally corresponds to solving the following minimization problem in our framework

$$\underset{V,\theta}{\arg\min}\, L_{\text{appearance}}(\mathcal{G}, \theta; \mathcal{I}, \mathcal{M}) + L_{\text{geometry}}(\mathcal{G}), \tag{2}$$

where $L_{\text{appearance}}$ compares the rendered appearance of the estimated surface to the camera images and L_{geometry} regularizes the mesh to avoid undesired vertex configurations.

3.2.1 Appearance

The appearance objective is composed of two terms

$$L_{\text{appearance}} = L_{\text{shading}} + L_{\text{silhouette}}, \tag{3}$$

where the shading term

$$L_{\text{shading}} = \lambda_{\text{shading}} \frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} \|I_i - \tilde{I}_i\|_1 \tag{4}$$

ensures that the color images produced by the shader $\tilde{I}_i$ correspond to the input images and the silhouette term

$$L_{\text{silhouette}} = \lambda_{\text{silhouette}} \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} \|M_i - \tilde{M}_i\|_1 \tag{5}$$

ensures that the rendered masks $\tilde{M}_i$ match the input masks for all views. Here, $\|\cdot\|_1$ denotes the mean absolute error of all pixels in an image. Formally, the masks $\tilde{M}_i$ are functions of the geometry $\mathcal{G}$ and the parameters of camera i, while the color images $\tilde{I}_i$ are also functions of the neural shader (or more precisely its parameters θ).

Separating the shading from the silhouette objective mainly has performance reasons: For a camera view i, the rasterization considers all pixels in the image, therefore computing the mask $\tilde{M}_i$ is cheap. However, shading is more involved and requires invoking the neural shader for all pixels after rasterization, which is an expensive operation. In practice, we only shade a subset of pixels inside the intersection of input and rendered masks while comparing the silhouette for all pixels. Additionally, we also limit the number of camera views considered in each gradient descent iteration.

3.2.2 Geometry Regularization

Naively moving the vertices unconstrained in each iteration quickly leads to undesirable meshes with degenerate triangles and self-intersections. We use a geometry regularization term that favors smooth solutions and is inspired by Luan *et al*. [36]:

$$L_{\text{geometry}} = L_{\text{laplacian}} + L_{\text{normal}}. \tag{6}$$

Let $V \in \mathbb{R}^{n \times 3}$ be a matrix with vertex positions as rows, the Laplacian term is defined as

$$L_{\text{laplacian}} = \lambda_{\text{laplacian}} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{\delta}_i\|_2^2, \tag{7}$$

where

$$\boldsymbol{\delta}_i = (LV)_i \in \mathbb{R}^3 \tag{8}$$

are the differential coordinates [1] of vertex i, $L \in \mathbb{R}^{n \times n}$ is the graph Laplacian of the mesh $\mathcal{G}$ and $\|\cdot\|_2$ is the Euclidean norm. Intuitively, by minimizing the magnitude of the differential coordinates of a vertex, we minimize its distance to the average position of its neighbors.

The normal consistency term is defined as

$$L_{\text{normal}} = \lambda_{\text{normal}} \frac{1}{|\bar{\mathcal{F}}|} \sum_{(i,j) \in \bar{\mathcal{F}}} (1 - \mathbf{n}_i \cdot \mathbf{n}_j)^2, \tag{9}$$

where $\bar{\mathcal{F}}$ is the set of triangle pairs that share an edge and $\mathbf{n}_i \in \mathbb{R}^3$ is the normal of triangle i (under an arbitrary ordering of the triangles). It computes the cosine similarity between neighboring face normals and enforces additional smoothness.

While some prior work (*e.g.*, [32, 36]) uses El Topo [4] for robust mesh evolution, we found that our geometric regularization sufficiently avoids degenerate vertex configurations. Without El Topo, we are unable to handle topology changes but avoid its impact on runtime performance.

Figure 4. Qualitative comparison on the DTU dataset. Left: Reference geometry and reconstruction results. Right: Point-to-mesh distance between the reference scan and the reconstructed mesh.

3.3. Optimization

Our optimization starts from an initial mesh that is computed from the masks and resembles a visual hull [28]. Alternatively, it can start from a custom mesh.

Similar to prior work, we use a coarse-to-fine scheme for the geometry: Starting from a coarsely triangulated mesh, we progressively increase its resolution during optimization. Inspired by Nicolet *et al.* [40] we remesh the surface with the method of Botsch and Kobbelt [3], halving the average edge length multiple times at fixed iterations. After mesh upsampling, we also increase the weights of the regularization terms by 4 and decrease the gradient descent step size for the vertices by 25 %, which we empirically found helps to improve the smoothness for highly tesselated meshes.

Since some quantities in geometry regularization (*e.g.* graph Laplacian) only depend on the connectivity of the mesh, we save time by precomputing them once after upsampling and reusing them in the iterations after.

4. Experimental Results

We implemented our method on top of the automatic differentiation framework PyTorch [45] and use the ADAM [24] optimizer for momentum-based gradient descent. Our differentiable rendering pipeline uses the high-performance primitives by Laine *et al.* [26]. In our experiments, we run 2000 gradient descent iterations and remesh after 500, 1000, and 1500 iterations. We randomly select one view per iteration to compute the appearance term and shade 75 % of mask pixels. The individual objective terms are weighted with $\lambda_{\text{shading}} = 1$, $\lambda_{\text{silhouette}} = 2$, $\lambda_{\text{laplacian}} = 40$, and $\lambda_{\text{normal}} = 0.1$. All time measurements were taken on a Windows workstation with an Intel Xeon 32×2.1 GHz CPU, 128 GB of RAM, and an NVIDIA Titan RTX GPU with 24 GB of VRAM.

Table 1. Quantitative results for multi-view reconstruction of objects from the DTU dataset. Chamfer scores are in *millimeters* and the COLMAP runtime is for the trim7 configuration.

Scan	COLMAP [48, 49] trim7 (trim0)		IDR [64]		NDS (Ours)	
	↓ Chamfer-L1	↓ time [min]	↓ Chamfer-L1	↓ time [min]	↓ Chamfer-L1	↓ time [min]
24	**0.45** (0.81)	66.81	1.58	551.83	4.24	**12.24**
37	**0.90** (2.03)	81.19	2.06	566.13	5.25	**7.56**
40	**0.36** (0.75)	65.47	0.75	550.19	1.30	**10.69**
55	**0.36** (1.20)	64.71	0.43	565.27	0.53	**7.54**
63	**0.90** (1.75)	75.62	1.06	553.07	2.47	**6.18**
65	0.94 (1.55)	62.62	**0.79**	568.41	1.22	**9.95**
69	**0.53** (1.02)	86.77	0.68	557.58	1.35	**9.64**
83	**1.16** (3.03)	77.96	1.38	745.18	1.59	**4.76**
97	**1.08** (1.42)	50.89	1.17	743.11	2.77	**7.32**
105	**0.63** (1.96)	51.59	0.88	742.31	1.15	**7.00**
106	**0.48** (0.99)	108.01	0.63	735.23	1.02	**7.41**
110	**0.58** (1.33)	85.87	0.99	752.94	3.18	**6.68**
114	**0.31** (0.50)	88.39	0.37	730.50	0.62	**7.99**
118	**0.44** (0.78)	105.75	0.50	748.60	1.65	**6.77**
122	**0.43** (1.17)	80.80	0.52	747.90	0.91	**5.76**
mean	**0.64** (1.35)	76.83	0.92	657.22	1.95	**7.83**

4.1. 3D Reconstruction

We demonstrate that our method can be used for multi-view 3D reconstruction. Starting from a coarse visual hull-like mesh, it quickly converges to a reasonable estimate of the object surface. We test our method on the DTU multi-view dataset [22] with the object selection and masks from previous work [41, 64]. We compare our results to two methods: (1) COLMAP [48, 49], a traditional SfM pipeline that serves as a baseline, and (2) IDR [64], a state-of-the-art analysis-by-synthesis method that uses neural signed distance functions as geometry representation. By default, our COLMAP results include trimming (trim7) and we indicate untrimmed results explicitly (trim0).

Figure 4 shows qualitative results for two objects from the DTU dataset and Table 1 contains quantitative results for all objects. We used the official DTU evaluation script to generate the Chamfer-L1 scores and benchmarked all workflows for the time measurements (including data loading times). For IDR and our method we disabled any intermediate visualizations.

In absolute terms (millimeters), the reconstruction accuracy of our method is close to both the traditional baseline and the state-of-the-art neural method. Although COLMAP reconstructs many surfaces accurately, only IDR and our method properly handle regions dominated by view-dependent effects (*e.g.* non-Lambertian materials) and produce watertight surfaces that can remain untrimmed. When COLMAP is used without trimming, the reconstruction becomes more complete but it is less accurate than ours for some objects. Our method is limited by the topology and genus of the initial mesh and therefore cannot capture some geometric details that can be recovered with more flexible surface representations. We also observe that our surfaces are often not as smooth as IDR's and concave regions are not as prominent. The latter is potentially related to the mesh stiffness induced by our geometry regularization.

On the other hand, our method is significantly faster: roughly 10 times faster than COLMAP and 80 times faster than IDR in the default configuration. Since the number of iterations is a hyper parameter for IDR and our method (and also has different semantics), we show equal time results for a fair comparison of both (Figure 5). Our method quickly converges to a detailed estimate, while IDR only recovers a rough shape in the same time. Even after 50 minutes, IDR still lacks details present in our result.

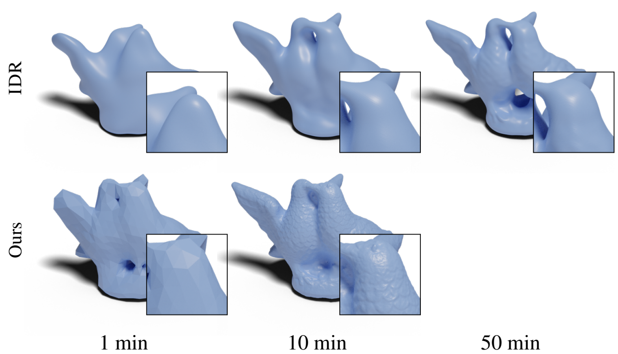

Figure 5. Equal time reconstruction. We consider our result converged after 10 minutes. Even after 50 minutes, IDR lacks details that are present in our reconstruction (*e.g.*, feathers and eyes).

Since IDR and our method have similar architectures, *i.e.*, both perform analysis-by-synthesis and use gradient descent to jointly optimize shape and appearance, we can meaningfully compare the runtime in more detail. Table 2 shows the runtime of *one* gradient descent iteration (see the supplementary material for a full decomposition of our runtime).

An iteration in IDR takes roughly twice the time as in our method, with the majority of time spent for ray marching the implicit function. In contrast, the time taken for rasterizing the triangle mesh is negligible in our method. Most of the ray marching time in IDR can be attributed to evaluating the network, thus switching to a more shallow neural representation could be a way to reduce the runtime. The remaining time is spent on operations like root finding, which could be accelerated by a more optimized implementation.

Table 2. Average runtime of one gradient descent iteration of IDR and our method, decomposed hierarchically. Geometry rendering time excludes shading and corresponds to ray marching 2048 pixels for IDR and rasterizing to 1.9 million pixels in our case.

	IDR		NDS (Ours)	
	time [s]	share	time [s]	share
Gradient descent iteration	0.3577	100 %	0.1561	100 %
↳ Geometry rendering	0.2099	↳ 58.7 %	0.0034	↳ 2.2 %
↳ SDF evaluation	0.1472	↳ 70.1 %	–	

However, the runtime difference in gradient descent iterations cannot be the sole reason for our fast convergence. Even though iterations in IDR require twice the time, it requires more than twice the total time to show the same level of detail as our method (see Figure 5).

In our method, we noticed that after mesh upsampling finer details quickly appear in the geometry. Thus, our fast convergence time might partially be related to the fact that we can locally increase the geometric freedom with a finer tessellation, while IDR and similar methods have no *explicit* control over geometric resolution.

4.2. Mesh Refinement

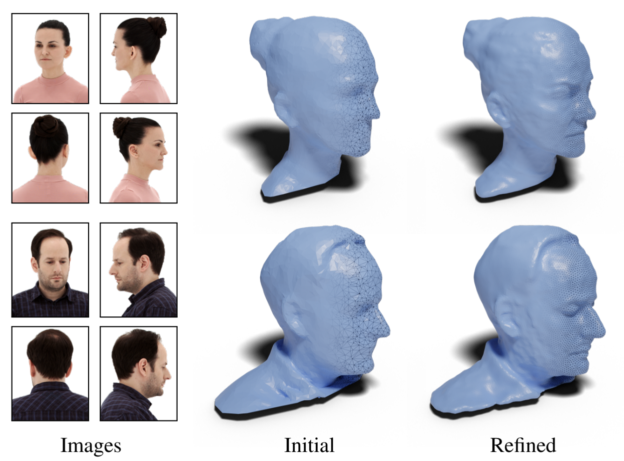

Figure 6. Refining meshes from an established multi-view reconstruction pipeline. Four of the 32 images are shown.

Many reconstruction workflows based on photo-consistency are very mature, established and deliver high-quality results. Yet, they can fail for challenging materials or different light conditions across images, producing smoothed outputs as a compromise to errors in photo-consistent matching.

Since our method can start from arbitrary triangle meshes, we propose the refinement of outputs from traditional pipelines as one possible application. Our method then acts as a *post-processing step*, improving the given geometry or parts of it with global multi-view constraints. We demonstrate this application on a human body dataset that contains 360° images of human subjects captured by 32 cameras and meshes from a traditional 3D reconstruction pipeline [50].

Figure 6 shows refinement results for the heads of two persons. We use 1000 iterations and upsample the mesh once. Since the initial mesh is already a good estimate of the true surface but the neural shader is randomly initialized, we rebalance the gradient descent step sizes so that the shader progresses faster than the geometry. We are able to recover details that are lost in the initial reconstruction. Very fine details like the facial hair are still challenging and lead to slight noise in our refined meshes.

4.3. Analysis of the Neural Shader

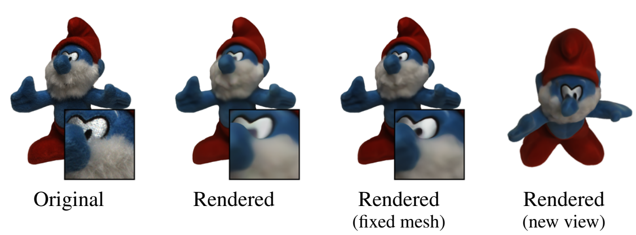

Figure 7. View synthesis for an input view and a new view. By fixing the mesh after reconstruction and continuing shader optimization, we can further improve the view synthesis results.

Although recovering geometry is the main focus of our work, investigating the neural shader after optimization can potentially give insights into the reconstruction procedure and the information that is encoded in the network.

Figure 7 shows that rendering with a trained shader can be used for basic view synthesis, producing reasonable results for input and novel views. Thus, the shader seems to learn a meaningful representation of appearance, disentangled from the geometry. If desired, the view synthesis quality can be further improved by continuing the optimization of the shader while keeping the mesh fixed.

The neural shader is a composite of two functions (Figure 3)

$$f_\theta(\mathbf{x}, \mathbf{n}, \boldsymbol{\omega}_o) = c\big(h(\mathbf{x}), \mathbf{n}, \boldsymbol{\omega}_o\big), \tag{10}$$

where $h : \mathbb{R}^3 \to \mathbb{R}^{256}$ transforms points in 3D space to positional features and $c : (\mathbb{R}^{256} \times \mathbb{R}^3 \times \mathbb{R}^3) \to [0,1]^3$ then extracts view-dependent colors. Both functions are dependent on parameters θ_h and θ_c, respectively. To further decompose the behavior of the shader, we perform a principle component analysis of the positional features from h

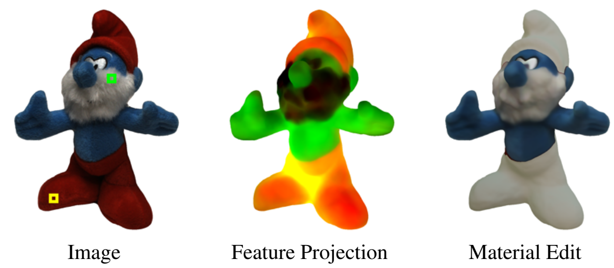

Figure 8. A principle component analysis of the shader's positional features reveals a connection to materials. We show the projection to the two largest principle components. Replacing features in the positional latent space before computing the view-dependent color enables simple material editing. The feature vector at the yellow square (pants material) determines regions that are replaced with the feature vector at the green square (beard material).

and project them to the two dominant components (Figure 8). The shader naturally learns similar positional features for regions with similar material, despite their distance in space. Variations in illumination also seem to be encoded in the positional features because shadowed regions have slightly different features than exposed regions of the same material.

We investigate the behavior of the view-dependent part by replacing feature vectors in the positional latent space and then extracting colors with the function c. More specifically, we replace all features representing one material by a feature representing another material (Figure 8). The results suggest that the function c reasonably generalizes to view and normal directions not encountered in combination with the replacement feature, which in this example leads to geometric features of the mesh being still perceivable and thus allows simple material editing.

4.4. Ablation Studies

We experiment with different encodings and network sizes for the position dependent part of the neural shader. In Figure 9, we show the different results using positional encoding (PE) [55], Gaussian Fourier features (GFF) [55], sinusoidal activations (SIREN) [52] and a standard MLP with ReLU activations.

While some of these methods can generate acceptable renderings, they do not necessarily guarantee a sharper geometry. In particular, we observe that although the image rendered with SIREN or GFF has sharp features, the geometric detail of the mesh is less adequate. A possible explanation is that the network might quickly overfit and compensate for geometrical inaccuracies only in the appearance. Conversely, finding the correct direction along which to move the mesh vertices might be more difficult without positional encoding. In our experiments, we have obtained accurate reconstructions using positional encoding with 4

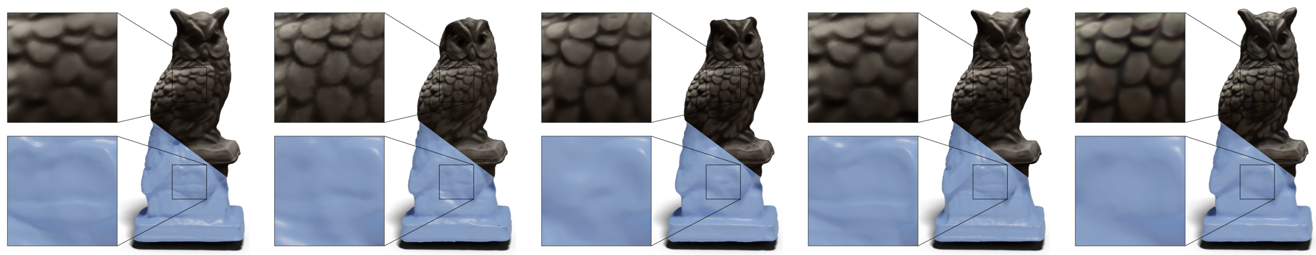

Figure 9. Ablation of different encodings for the positional features and activation functions. Note that the ears of the owl are only present in few views and not part of the reference geometry, thus are not considered in the comparison. For ReLU and SIREN we use no encoding.

octaves, as opposed to the recommended 10 [39].

We also examined the effect of different network sizes on the geometry and appearance (see supplementary). While there are no dramatic differences between the results, we observe that configurations with less than 2 layers or with more than 512 units per layer result in fewer geometric details. Additional ablation studies for the initial geometry and the objective function can also be found in the supplementary.

5. Concluding Remarks

We have presented a fast analysis-by-synthesis pipeline for 3D surface reconstruction from multiple images. Our method jointly optimizes a triangle mesh and a neural shader as representations of geometry and appearance to reproduce the input images, thereby combining the speed of triangle mesh optimization with the generality of neural rendering.

Our approach can match state-of-the-art methods in reconstruction accuracy, yet is significantly faster with average runtimes below 10 minutes.

Using triangle meshes as geometry representation makes our method fully compatible with many traditional reconstruction workflows. Thus, instead of replacing the complete reconstruction architecture, our method can be integrated as a part, *e.g.*, a refinement step.

Finally, a preliminary analysis of the neural shader suggests that it decomposes appearance in a natural way, which could help our understanding of neural rendering and enable simple ways for altering the scene appearance (*e.g.* for material editing).

Limitations and Future Work. While triangle meshes are a simple and fast representation, using them robustly in 3D reconstruction with differentiable rendering is still challenging. We currently avoid undesired mesh configurations (*e.g.* self-intersections) with carefully weighted geometry regularizers that steer the optimization towards smooth solutions. Finding an appropriate balance between smoothness and rendering terms is not always straightforward and can require fine-tuning for custom data. In this regard, we are excited about integrating very recent work that proposes gradient preconditioning instead of geometry regularization [40].

Topology changes are a challenge for most mesh-based methods, including ours, and computationally expensive to handle (*e.g.* with El Topo [4]). Likewise, adaptive reconstruction-aware subdivision would be preferable over standard remeshing, potentially including learned priors [31]. Instead of moving vertices directly, a (partially pretrained) neural network could drive the deformation [17,18], making it more efficient, detail-aware, and less dependent on geometric regularization.

While neural shading is a powerful component of our system, allowing us to handle non-Lambertian surfaces and complex illumination, it is also a major obstruction for interpretability. The effect of changes to the network architecture can only be evaluated with careful experiments and often the black box shader behaves in non-intuitive ways. We experimentally provided preliminary insights but also think that a more thorough analysis is needed. Alternatively, physical light transport models could be combined with more specialized neural components (*e.g.* irradiance or material) to isolate their effects. In this context, pretraining components to include learned priors also seems like a promising direction.

Although the shader can handle arbitrary material and light in theory, this claim needs more investigation. A possible path is exhaustive experiments with artificial scenes, starting with the simplest cases (how well does it handle a *perfectly* Lambertian surface?).

Acknowledgements

This work is part of the INVICTUS project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952147.

References

[1] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003. 4

[2] David C. Blumenthal-Barby and Peter Eisert. High-resolution depth for binocular image-based modeling. *Comput. Graph.*, 39:89–100, Apr. 2014. 2

[3] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, page 185–192, New York, NY, USA, 2004. Association for Computing Machinery. 5

[4] Tyson Brochu and Robert Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493, 2009. 4, 8

[5] Neill D. F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 766–779, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 2

[6] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4), July 2015. 2

[7] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: A vlsi system for high performance graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, page 21–30, New York, NY, USA, 1988. Association for Computing Machinery. 2

[8] Amael Delaunoy, Emmanuel Prados, Pau Gargallo I Piracés, Jean-Philippe Pons, and Peter Sturm. Minimizing the Multi-view Stereo Reprojection Error for Triangular Surface Meshes. In Mark Everingham, Chris J. Needham, and Roberto Fraile, editors, *BMVC 2008 - British Machine Vision Conference*, pages 1–10, Leeds, United Kingdom, Sept. 2008. BMVA. 2

[9] Frank Dellaert and Yen-Chen Lin. Neural volume rendering: Nerf and beyond. *CoRR*, abs/2101.05204, 2021. 2

[10] C.H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 46–53, 2003. 2

[11] Pascal Fua and Yvan G Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56, 1995. 2

[12] Yasutaka Furukawa and Jean Ponce. Carved visual hulls for image-based modeling. volume 81, pages 564–577, 07 2006. 2

[13] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010. 2

[14] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, 2015. 2

[15] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, Danhang Tang, Anastasia Tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Paul Debevec, and Shahram Izadi. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Trans. Graph.*, 38(6), Nov. 2019. 2

[16] X. Han, H. Laga, and M. Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(05):1578–1604, may 2021. 1

[17] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Trans. Graph.*, 38(4), July 2019. 8

[18] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), July 2020. 8

[19] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-Driven Automatic 3D Model Simplification. In Adrien Bousseau and Morgan McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021. 2

[20] Vu Hoang Hiep, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards high-resolution large-scale multi-view stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1437, 2009. 2

[21] Isidro and Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1335–1342 vol.2, 2003. 2

[22] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 5

[23] Petr Kellnhofer, Lars Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *CVPR*, 2021. 3

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[25] Hamid Laga, Laurent Valentin Jospin, F. Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2020. 1

[26] 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. 5

[27] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. *arXiv:2004.07484*, 2020. 2

[28] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. 5

[29] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 1, 3

[30] Chen-Hsuan Lin, Oliver Wang, Bryan C. Russell, Eli Shechtman, Vladimir G. Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[31] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Trans. Graph.*, 39(4), July 2020. 8

[32] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. *ACM Transactions on Graphics*, 2018. 2, 4

[33] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8293–8304, 2019. 1

[34] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 1

[35] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.*, 38(6), Nov. 2019. 1, 2

[36] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum*, 2021. 1, 2, 4

[37] Jiahui Lyu, Bojian Wu, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Differentiable refraction-tracing for mesh reconstruction of transparent objects. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2020)*, 39(6):195:1–195:13, 2020. 1, 2

[38] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[39] 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, 8

[40] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM SIGGRAPH Asia 2021*, 2021. 5, 8

[41] 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, 3, 5

[42] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 38(6), Nov. 2019. 1, 3

[43] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021. 1, 3

[44] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 3

[45] 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. 5

[46] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 3

[47] Alyn P Rockwood and Jim Winget. Three-dimensional object reconstruction from two-dimensional images. *Computer-Aided Design*, 29(4):279–285, 1997. Reverse Engineering of Geometric Models. 2

[48] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[49] 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. 5

[50] 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. 2, 7

[51] Steven Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. volume 1, pages 519–528, 01 2006. 2

[52] 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. 7

[53] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007. 2

[54] R. Tamura, S. Ito, N. Kaneko, and K. Sumi. Towards detailed 3d modeling: Mesh super-resolution via deformation. In *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6, Los Alamitos, CA, USA, aug 2020. IEEE Computer Society. 2

[55] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 3, 7

[56] Ayush Tewari, O Fried, J Thies, V Sitzmann, S Lombardi, Z Xu, T Simon, M Nießner, E Tretschk, L Liu, et al. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, pages 1–320. 2021. 2

[57] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), July 2019. 3

[58] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012. 2

[59] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, New York, NY, USA, 2009. Association for Computing Machinery. 2

[60] Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, 2012. 2

[61] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 3

[62] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *ICCV*, 2019. 2

[63] Chenglei Wu, Bennett Wilburn, Yasuyuki Matsushita, and Christian Theobalt. High-quality shape from multi-view stereo and shading under general illumination. In *CVPR 2011*, pages 969–976, 2011. 2

[64] 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, 3, 4, 5

[65] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 1

[66] Tianli Yu, Ning Xu, and Narendra Ahuja. Shape and view independent reflectance map from multiple views. In Tomás Pajdla and Jiri Matas, editors, *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV*, volume 3024 of *Lecture Notes in Computer Science*, pages 602–616. Springer, 2004. 2

[67] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4), July 2020. 1

[68] Li Zhang and Steven M. Seitz. Image-based multiresolution shape recovery by surface deformation. In Sabry F. El-Hakim and Armin Gruen, editors, *Videometrics and Optical Methods for 3D Shape Measurement*, volume 4309, pages 51 – 61. International Society for Optics and Photonics, SPIE, 2000. 2