This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Extracting Triangular 3D Models, Materials, and Lighting From Images

Jacob Munkberg[1] Jon Hasselgren[1] Tianchang Shen[1,2,3] Jun Gao[1,2,3]
Wenzheng Chen[1,2,3] Alex Evans[1] Thomas Müller[1] Sanja Fidler[1,2,3]

[1]NVIDIA [2]University of Toronto [3]Vector Institute

Abstract

We present an efficient method for joint optimization of topology, materials and lighting from multi-view image observations. Unlike recent multi-view reconstruction approaches, which typically produce entangled 3D representations encoded in neural networks, we output triangle meshes with spatially-varying materials and environment lighting that can be deployed in any traditional graphics engine unmodified. We leverage recent work in differentiable rendering, coordinate-based networks to compactly represent volumetric texturing, alongside differentiable marching tetrahedrons to enable gradient-based optimization directly on the surface mesh. Finally, we introduce a differentiable formulation of the split sum approximation of environment lighting to efficiently recover all-frequency lighting. Experiments show our extracted models used in advanced scene editing, material decomposition, and high quality view interpolation, all running at interactive rates in triangle-based renderers (rasterizers and path tracers).

1. Introduction

3D content creation is a challenging, mostly manual task which requires both artistic modeling skills and technical knowledge. Efforts to automate 3D modeling can save substantial production costs or allow for faster and more diverse content creation. Photogrammetry [48,56] is a popular technique to assist in this process, where multiple photos of an object are converted into a 3D model. Game studios leverage photogrammetry to quickly build highly detailed virtual landscapes [21]. However, this is a multi-stage process, including multi-view stereo [52] to align cameras and find correspondences, geometric simplification, texture parameterization, material baking and delighting. This complex pipeline has many steps with conflicting optimization goals and errors that propagate between stages. Artists rely on a plethora of software tools and significant manual adjustments to reach the desired quality of the final 3D model.

Project page: https://nvlabs.github.io/nvdiffrec/

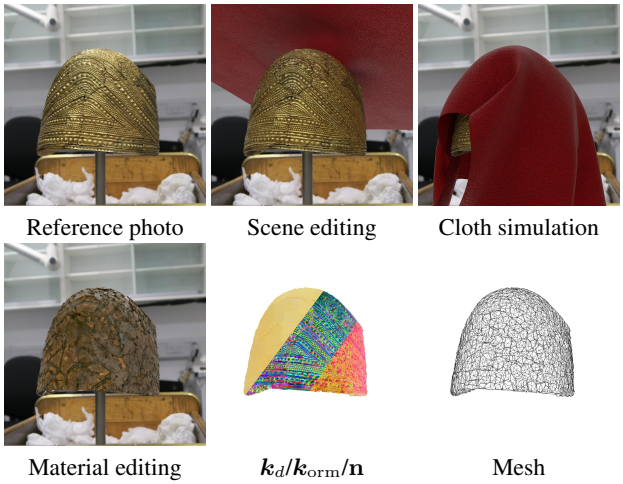


Figure 1. We reconstruct a triangular mesh with unknown topology, spatially-varying materials, and lighting from a set of multi-view images. We show examples of scene manipulation using off-the-shelf modeling tools, enabled by our reconstructed 3D model.

Our goal is to frame this process as an *inverse rendering* task, and optimize as many steps as possible jointly, driven by the quality of the rendered images of the reconstructed model, compared to the captured input imagery. Recent work approaches 3D reconstruction with neural rendering, and provides high quality novel view synthesis [39]. However, these methods typically produce representations that entangle geometry, materials and lighting into neural networks, and thus cannot easily support scene editing operations. Furthermore, to use them in traditional graphics engines, one needs to extract geometry from the network using methods like Marching Cubes which may lead to poor surface quality, particularly at low triangle counts. Recent neural methods can disentangle shape, materials, and lighting [3, 68, 70], but sacrifice reconstruction quality. Also, the materials encoded in neural networks cannot easily be edited or extracted in a form compatible with traditional game engines. In contrast, we reconstruct 3D content compatible with traditional graphics engines, supporting relighting and scene editing.

In this paper, we present a highly efficient inverse render-

ing method capable of extracting triangular meshes of unknown topology, with spatially-varying materials and lighting from multi-view images. We assume that the object is illuminated under one unknown environment lighting condition, and that we have corresponding camera poses and masks indicating the object in these images, as in past work [3]. Our approach learns topology and vertex positions for a surface mesh without requiring any initial guess for the 3D geometry. The heart of our method is a differentiable surface model based on a deformable tetrahedral mesh [54], which we extend to support spatially-varying materials and high dynamic range (HDR) environment lighting, through a novel differentiable split sum approximation. We optimize geometry, materials and lighting (50M+ parameters) jointly using a highly optimized differentiable rasterizer with deferred shading [22, 30]. The resulting 3D model can be deployed without conversion on any device supporting triangle rendering, including phones and web browsers, and renders at interactive rates.

Experiments show our extracted models used in scene editing (e.g., Figure 1), material decomposition, and high quality view interpolation, all running at interactive rates in triangle-based renderers (rasterizers and path tracers).

2. Related Work

2.1. Multi-view 3D Reconstruction

Classical methods for multi-view 3D reconstruction either exploit inter-image correspondences [1, 11, 12, 52] to estimate depth maps or use voxel grids to represent shapes [10, 53]. The former methods typically fuse depth maps into point clouds, optionally generating meshes [27]. They rely heavily on the quality of matching, and errors are hard to rectify during post-processing. The latter methods estimate occupancy and color for each voxel and are often limited by the cubic memory requirement.

Neural implicit representations leverage differentiable rendering to reconstruct 3D geometry with appearance from image collections [24, 39, 43]. NeRF [39] and follow-ups [15, 38, 41, 42, 49, 50, 60, 62, 66, 69], use volumetric representations and compute radiance by ray marching through a neurally encoded 5D light field. While achieving impressive results on novel view synthesis, geometric quality suffers from the ambiguity of volume rendering [69]. Surface-based rendering methods [43, 64] use implicit differentiation to obtain gradients, optimizing the underlying surface directly. Unisurf [47] is a hybrid approach that gradually reduces the sampling region, encouraging a volumetric representation to converge to a surface, and NeuS [59] provides an unbiased conversion from SDF into density for volume rendering. Common for all methods is that they rely on ray marching for rendering, which is computationally expensive both during training and inference. While implicit surfaces can be converted to meshes for fast inference, this introduces additional error not accounted for during optimization [54]. We optimize the end-to-end image loss of an explicit mesh representation, supporting intrinsic decomposition of shape, materials and lighting by design, and utilizing efficient differentiable rasterization [30].

Explicit surface representations are proposed to estimate explicit 3D mesh from images [7, 8, 14, 23, 34, 35, 54]. Most approaches assume a given, fixed mesh topology [7, 8, 23, 35], but this is improved in recent work [14, 34, 54]. In particular, DMTet [54] directly optimizes the surface mesh using a differentiable marching tetrahedral layer. However, it focuses on training with 3D supervision. In this work, we extend DMTet to 2D supervision, using differentiable rendering to jointly optimize topology, materials, and lighting.

2.2. BRDF and Lighting Estimation

Beyond geometry, several techniques estimate surface radiometric properties from images. Previous work on BTF and SVBRDF estimation rely on special viewing configurations, lighting patterns or complex capturing setups [2, 5, 16–18, 20, 31, 51, 61]. Recent methods exploit neural networks to predict BRDF from images [13, 19, 32, 33, 37, 44]. Differentiable rendering methods [7, 8, 22, 35, 71] learn to predict geometry, SVBRDF and, in some cases, lighting via 2D image loss. Still, their shape is generally deformed from a sphere and cannot represent arbitrary topology.

Neural implicit representations successfully estimate lighting and BRDF from image collections. Bi et al. [2] and NeRV [57] model light transport to support advanced lighting effects, e.g., shadows, but have very high computational cost. Most related to our work are neural 3D reconstruction methods for jointly estimating shape, BRDFs and lighting from images [3, 4, 68, 70], while providing an *intrinsic decomposition* of these terms. Illumination is represented using mixtures of spherical Gaussians (NeRD [3], PhySG [68]), or low resolution envmaps (NeRFactor [70]), in both cases limited to low frequency illumination. In contrast, we propose a differentiable split sum lighting model, also adopted by the concurrent work Neural-PIL [4]. These neural implicit methods use multiple MLPs to factorize

Method	Geometry	Factorize	Training	Inference
NeRF [39]	NV		day	seconds
NGP-NeRF [41]	NV		minutes	ms
NeRD [3]	NV	✓	days	seconds
NerFactor [70]	NV	✓	days	seconds
PhySG [68]	NS	✓	day	seconds
NeuS [59]	NS		day	seconds
Our	Mesh	✓	hour	ms

Table 1. Taxonomy of methods. NV: Neural volume, NS: Neural surface. Factorize indicates if the method supports some decomposition of geometry, materials, and lighting.

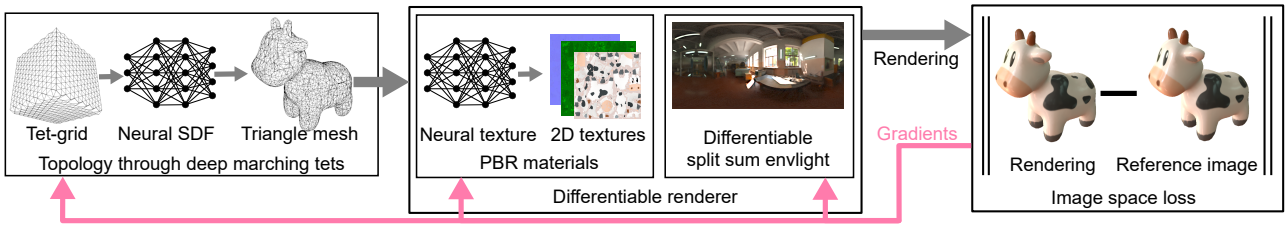


Figure 2. **Overview of our approach.** We learn topology, materials, and environment map lighting jointly from 2D supervision. We leverage differentiable marching tetrahedrons to directly optimize topology of a triangle mesh. While the topology is drastically changing, we learn materials through volumetric texturing, efficiently encoded using an MLP with positional encoding. Finally, we introduce a differentiable version of the split sum approximation for environment lighting. Our output representation is a triangle mesh with spatially varying 2D textures and a high dynamic range environment map, which can be used unmodified in standard game engines. The system is trained end-to-end, supervised by loss in image space, with gradient-based optimization of all stages. Spot model by Keegan Crane.

the representation, resulting in long training and inference times. Furthermore, these methods forgo the vast ecosystem of available 3D modeling and rendering tools, "reinventing the wheel" for tasks such as rendering, scene editing [63] and simulation. In contrast, our output is directly compatible with existing renderers and modeling tools. We optimize an explicit surface mesh, BRDF parameters, and lighting stored in an HDR probe, achieving faster training speed and better decomposition results. Table 1 shows a high level comparison of the methods.

3. Our Approach

We present a method for 3D reconstruction supervised by multi-view images of an object illuminated under one unknown environment lighting condition, together with known camera poses and background segmentation masks. The target representation consists of triangle meshes, spatially-varying materials (stored in 2D textures) and lighting (a high dynamic range environment probe). We carefully design the optimization task to *explicitly* render triangle meshes, while robustly handling arbitrary topology. Hence, unlike most recent work using neural implicit surface or volumetric representations, we directly optimize the target shape representation.

Concretely, we adapt Deep Marching Tetrahedra [54] (DMTet) to work in the setting of 2D supervision, and jointly optimize shape, materials, and lighting. At each optimization step, the shape representation – parameters of a signed distance field (SDF) defined on a grid with corresponding per-vertex offsets – is converted to a triangular surface mesh using a marching tetrahedra layer. Next, we render the extracted surface mesh in a differentiable rasterizer with deferred shading, and compute loss in image space on the rendered image compared to a reference image. Finally, the loss gradients are back-propagated to update the shape, textures and lighting parameters. Our approach is summarized in Figure 2 and each step is described in detail below; Section 3.1 outlines our topology optimization, Section 3.2 presents the spatially-varying shading model, and our approach for reconstructing all-frequency environment lighting is described in Section 3.3.

Optimization task Let ϕ denote our optimization parameters (i.e., SDF values and vertex offsets representing the shape, spatially varying material and light probe parameters). For a given camera pose, c, the differentiable renderer produces an image $I_\phi(c)$. The reference image $I_{\text{ref}}(c)$ is a view from the same camera. Given a loss function L, we minimize the empirical risk

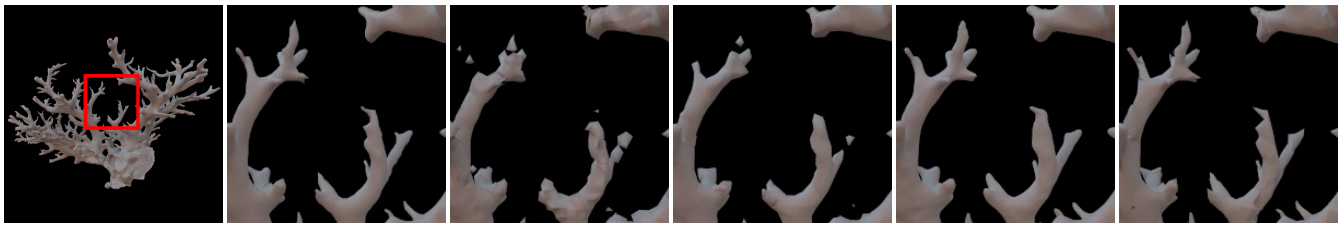
$$\underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_c\big[L\big(I_\phi(c), I_{\text{ref}}(c)\big)\big] \tag{1}$$

using Adam [28] based on gradients w.r.t. the optimization parameters, $\partial L/\partial\phi$, which are obtained through differentiable rendering. Our renderer uses physically based shading and produces images with high dynamic range. Therefore, the objective function must be robust to the full range of floating-point values. Following recent work in differentiable rendering [22], our loss function is $L = L_{\text{image}} + L_{\text{mask}} + \lambda L_{\text{reg}}$, an image space loss, L_{image} (L_1 norm on tone mapped colors), a mask loss, L_{mask} (squared L_2) and a regularizer L_{reg} (Equation 2) to improve geometry. Please refer to the supplemental material for details.

Assumptions For performance reasons we use a differentiable rasterizer with deferred shading [22], hence reflections, refractions (e.g., glass), and translucency are not supported. During optimization, we only renderer direct lighting without shadows. Our shading model uses a diffuse Lambertian lobe and a specular, isotropic microfacet GGX lobe, which is commonly used in modern game engines [26, 29]. Both dielectrics and metal materials are supported. We note that our approach directly generalizes to a differentiable path tracer [45, 46], but at a significantly increased computational cost.

3.1. Learning Topology

Volumetric and implicit shape representations (e.g., SDFs) can be converted to meshes through Marching



	Reference	NeRF (57k tris)	NeuS (53k tris)	NeuS (900k tris)	Our (53k tris)
Chamfer L_1 $\times 10^{-4}$		33.4	9.19	5.84	4.65

Figure 3. Triangle mesh extraction from a set of 256 rendered images w/ masks. Damicornis model from the Smithsonian 3D repository [55], We extracted meshes from NeRF and NeuS using Marching Cubes for a target triangle count of roughly 50k triangles and optimized the example in our pipeline for a similar count. We show renderings of the extracted meshes in a path tracer and report the Chamfer loss. We note that NeuS, which optimizes a surface representation, significantly improves on the volumetric representation used by NeRF for this example. Furthermore, our end-to-end optimization of a triangle mesh improves both the visual quality and the Chamfer loss at a fixed triangle count. When drastically increasing the triangle count in the NeuS mesh extraction (from 53k to 900k triangles), the quality improves significantly, indicating that NeuS has a high quality internal surface representation. Still, our mesh with 53k triangles is on par with the high resolution NeuS output, indicating the benefit of directly optimizing the mesh representation.

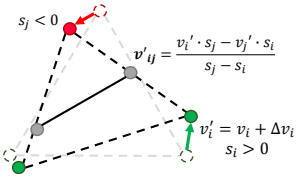


Figure 4. Marching Tetrahedra extracts faces from a tetrahedral grid with grid vertices, $v_i' = v_i + \Delta v_i$ and scalar SDF values, s_i. For tets with $\text{sign}(s_i) \neq \text{sign}(s_j)$, faces are extracted, and the face vertices, v_{ij}, are determined by by linear interpolation.

Cubes [36] (MC) in a post-processing step. However, MC inevitably imposes discretization errors. As a result, the output mesh quality, particularly at the moderate triangle counts typically used in real-time rendering, is often not sufficient. Similarly, simplifying dense extracted meshes using decimation tools may introduce errors in rendered appearance. To avoid these issues, we explicitly render triangle meshes during optimization. We leverage Deep Marching Tetrahedra [54] (DMTet) in a 2D supervision setting through differentiable rendering. DMTet is a hybrid 3D representation that represents a shape with a discrete SDF defined on vertices of a deformable tetrahedral grid. The SDF is converted to triangular mesh using a differentiable marching tetrahedra layer (MT), as illustrated in Figure 4. The loss, in our case computed on renderings of the 3D model, is back-propagated to the implicit field to update the surface topology. This allows us to directly optimize the surface mesh and rendered appearance *end-to-end*.

We illustrate the advantage of end-to-end learning in Figure 3, where we compare our meshes to those generated by competing methods. While NeRF [39] (volumetric rep.) and NeuS [59] (implicit surface rep.) provide high quality view interpolation, the quality loss introduced in the MC step is significant at low triangle counts.

Given a tetrahedral grid with vertex positions v, DMTet learns SDF values, s, and deformation vectors Δv. The SDF values and deformations can either be stored explicitly as values per grid vertex, or implicitly [43, 47] by a neural network. At each optimization step, the SDF is first converted to a triangular surface mesh using MT, which is shown to be differentiable w.r.t. SDF and can change surface topology in DMTet [54]. Next, the extracted mesh is rendered using a differentiable rasterizer to produce an output image, and image-space loss gradients are back-propagated to the SDF values and offsets (or network weights). A neural SDF representation can act as a smoothness prior, which can be beneficial in producing well-formed shapes. Directly optimizing per-vertex attributes, on the other hand, can capture higher frequency detail and is faster to train. In practice, the optimal choice of parametrization depends on the ambiguity of geometry in multi-view images. We provide detailed analysis in the supplemental materials.

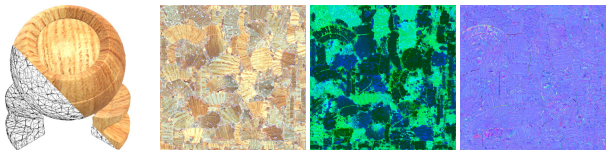
To reduce floaters and internal geometry, we regularize the SDF values of DMTet similar to Liao et al. [34]. Given the binary cross-entropy H, sigmoid function σ, and the sign function $\text{sign}(x)$, we define the regularizer as

$$L_{\text{reg}} = \sum_{i,j \in \mathbb{S}_e} H\left(\sigma(s_i), \text{sign}\left(s_j\right)\right) + H\left(\sigma(s_j), \text{sign}\left(s_i\right)\right), \quad (2)$$

where we sum over the set of unique edges, $\mathbb{S}_e$, in the tetrahedral grid, for which $\text{sign}(s_i) \neq \text{sign}(s_j)$. Intuitively, this reduces the number of sign flips and simplifies the surface, penalizing internal geometry or floaters. We ablate the choice of regularization loss in the supplemental material.

3.2. Shading Model

Material Model We follow previous work in differentiable rendering [22] and use the physically-based (PBR) material model from Disney [6]. This lets us easily import game assets and render our optimized models directly in existing engines without modifications. This material model combines a diffuse term with an isotropic, specular GGX lobe [58]. Referring to Figure 5, the diffuse lobe param-



wire/shaded $\boldsymbol{k}_d$ $\boldsymbol{k}_{\text{orm}}$ normals

Figure 5. We represent 3D models as a triangular mesh and a set of spatially varying materials following a standard PBR model.

eters $\boldsymbol{k}_d$ are provided as a four-component texture, where the optional fourth channel α represents transparency. The specular lobe is described by a roughness value, r, for the GGX normal distribution function and a metalness factor, m, which interpolates between plastic and metallic appearance by computing a specular highlight color according to $\boldsymbol{k}_s = (1-m)\cdot 0.04 + m\cdot \boldsymbol{k}_d$ [26]. Following a standard convention, we store these values in a texture $\boldsymbol{k}_{\text{orm}} = (o, r, m)$, where o is left unused. Finally, we include a tangent space normal map, $\mathbf{n}$, to capture high frequency shading detail. We regularize material parameters using a smoothness loss [70], please refer to our supplemental material for details.

Texturing Automatic texture parametrization for surface meshes is an active research area in computer graphics. We optimize topology, which requires continually updating the parametrization, potentially introducing discontinuities into the training process. To robustly handle texturing during topology optimization, we leverage volumetric texturing, and use world space position to index into our texture. This ensures that the mapping varies smoothly with both vertex translations and changing topology.

The memory footprint of volumetric textures grows cubically, which is unmanageable for our target resolution. We therefore extend the approach of PhySG [68], using a multilayer perceptron (MLP) to encode all material parameters in a compact representation. This representation can adaptively allocate detail near the 2D manifold representing the surface mesh, which is a small subset of the dense 3D volume. More formally, we let a positional encoding + MLP represent a mapping $\mathbf{x} \rightarrow (\boldsymbol{k}_d, \boldsymbol{k}_{\text{orm}}, \mathbf{n})$, e.g., given a world space position $\mathbf{x}$, compute the base color, $\boldsymbol{k}_d$, specular parameters, $\boldsymbol{k}_{\text{orm}}$ (roughness, metalness), and a tangent space normal perturbation, $\mathbf{n}$. We leverage the tiny-cuda-nn framework [40], which provides efficient kernels for hashgrid positional encoding [41] and MLP evaluations.

Once the topology and MLP texture representation have converged, we *re-parametrize* the model: we generate unique texture coordinates using xatlas [65] and sample the MLP on the surface mesh to initialize 2D textures, then continue the optimization with fixed topology. Referring to Figure 6, this effectively removes texture seams introduced by the (u, v)-parametrization, and may also increase texture detail as we can use high resolution 2D textures for each of

Figure 6. Sampling out the volumetric representation to create 2D textures results in texture seams (left). However, further optimization (right), quickly removes the seams automatically.

$\boldsymbol{k}_d$, $\boldsymbol{k}_{\text{orm}}$, and $\mathbf{n}$. This results in 2D textures compatible with standard 3D tools and game engines.

3.3. Image Based Lighting

We adopt an image based lighting model, where the scene environment light is given by a high-resolution cube map. Following the rendering equation [25], we compute the outgoing radiance $L(\omega_o)$ in direction ω_o by:

$$L(\omega_o) = \int_{\Omega} L_i(\omega_i) f(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i. \tag{3}$$

This is an integral of the product of the incident radiance, $L_i(\omega_i)$ from direction ω_i and the BSDF $f(\omega_i, \omega_o)$. The integration domain is the hemisphere Ω around the surface intersection normal, $\mathbf{n}$.

Below, we focus on the specular part of the outgoing radiance, where, in our case, the BSDF is a Cook-Torrance microfacet specular shading model [9] according to:

$$f(\omega_i, \omega_o) = \frac{D\,G\,F}{4(\omega_o \cdot \mathbf{n})(\omega_i \cdot \mathbf{n})}, \tag{4}$$

where D, G and F are functions representing the GGX [58] normal distribution (NDF), geometric attenuation and Fresnel term, respectively.

High quality estimates of image based lighting can be obtained by Monte Carlo integration. For low noise levels, large sample counts are required, which is typically too expensive for interactive applications. Thus, spherical Gaussians (SG) and spherical harmonics (SH) are common approximations for image based lighting [3, 7, 68]. They allow for control over the lighting frequency through varying the number of SG lobes (or SH coefficients), and are efficient representations for low to medium frequency lighting. However, representing high frequency and highly specular materials requires many SG lobes, which comes at a high runtime cost and hurts training stability.

We instead draw inspiration from real-time rendering, where the *split sum* approximation [26] is a popular, efficient method for all-frequency image based lighting. Here, the lighting integral from Equation 3 is approximated as:

$$L(\omega_o) \approx \int_{\Omega} f(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i \int_{\Omega} L_i(\omega_i) D(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i. \tag{5}$$

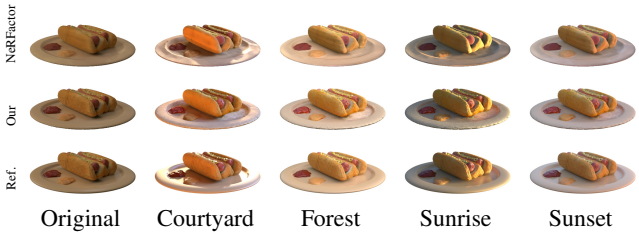


Figure 7. Relighting quality for a scene from the NeRFactor dataset, with our examples relit using Blender, and NeRFactor results generated using the public code.

The first term represents the integral of the specular BSDF with a solid white environment light. It only depends on the parameters $\cos\theta = \omega_i \cdot \mathbf{n}$ and the roughness r of the BSDF, and can be precomputed and stored in a 2D lookup texture. The second term represents the integral of the incoming radiance with the specular NDF, D. Following Karis [26], this term is also pre-integrated and represented by a filtered cubemap. In each mip-level, the environment map is integrated against D for a fixed roughness value (increased roughness at coarser mips).

The split sum approach is popular for its modest runtime cost, using only two texture lookups: query the 2D lookup texture representing the first term based on $(r, \cos\theta)$ and the mip pyramid at level r, in direction ω_o. This should be compared to evaluating SG products with hundreds of lobes for each shading point. Furthermore, it uses the standard GGX parametrization, which means that we can relight our extracted models with different kinds of light sources (point, area lights etc.) and use our reconstructed materials with no modifications in most game engines and modeling tools.

We introduce a *differentiable* version of the split sum shading model to learn environment lighting from image observations through differentiable rendering. We let the texels of a cube map (typical resolution $6 \times 512 \times 512$) be the trainable parameters. The base level represents the pre-integrated lighting for the lowest supported roughness value, and each smaller mip-level is constructed from the base level using the pre-filtering approach from Karis [26].

To obtain texel gradients, we express the lighting computations using PyTorch's auto-differentiation. However, the pre-filtering of the second term in Equation 5 must be updated in each training iteration, and therefore warrant a specialized CUDA implementation to reduce the training cost. This term can either be estimated through Monte-Carlo integration (BSDF importance sampling), or by pre-filtering the environment map in a solid-angle footprint derived from the NDF. To reduce noise, at the cost of introducing some bias, we chose the latter approach. Please refer to our supplemental material for implementation details.

We additionally create a single filtered low-resolution ($6 \times 16 \times 16$) cube map representing the *diffuse* lighting. The process is identical to the filtered specular probe, sharing the same trainable parameters, average-pooled to the mip level with roughness $r = 1$. The pre-filtering of the diffuse term only uses the cosine term, $\omega_i \cdot \mathbf{n}$. The two filtering steps are fully differentiable and are performed at each training step.

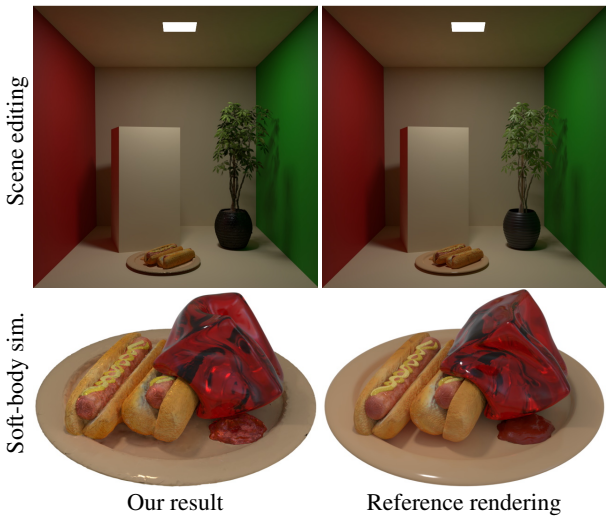


Figure 8. To highlight the benefits of our explicit representation, we insert two reconstructed models into the Cornell box. Note that the objects accurately interact with the scene lighting, and cast shadows (e.g., the green wall). Next, we use our reconstructed hotdog model in a soft-body physics simulation, dropping red jelly on the plate. We run the entire simulation (21 frames) on both the reference 3D mesh and our reconstructed mesh, and display the last frame. Note that these applications are not currently feasible for neural volumetric representations.

4. Experiments

In the following, we evaluate our system for a variety of applications. To emphasize our approach's explicit decomposition into a triangle mesh and materials, we show re-lighting, editing, and simulation using off-the shelf tools. We also compare to recent neural methods supporting factorization: NeRD [3] and NeRFactor [70]. While not our main focus, we include view interpolation results to establish a baseline comparison to state-of-the-art methods. Finally, we compare our split-sum approximation against spherical Gaussians for image-based lighting.

4.1. Scene Editing and Simulation

Our factorized scene representation enables advanced scene editing. Previous work using density-based neural representations only supports rudimentary relighting and simple forms of scene edits [3, 68, 70].

In Figure 7 we compare the relighting quality of our reconstructed model, rendered using the Blender Cycles path tracer, with the results of NeRFactor (rendered by evaluating a neural network). A quantitative summary is provided

	Relighting			k_d		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRFactor	23.78	0.907	0.112	23.11	0.917	0.094
Our	24.53	0.914	0.085	24.75	0.924	0.092

Table 2. Relighting quality for NeRFactor's synthetic dataset. The reported image metrics are the arithmetic mean over 8 validation views and 8 light probes for all 4 test scenes. We also show metrics for the k_d (albedo) textures. Following NeRFactor, we note that the scale factor between material and lighting is indeterminable and therefore normalize albedo images by the average intensity of the reference before measuring errors.

in Table 2, where we also measure the quality of the reconstructed albedo textures. We note that our method produces more detailed results and outperforms NeRFactor in all metrics. Our artifacts come mainly from the mismatch between training (using a rasterizer), and inference (using full global illumination). In areas with strong shadowing or color bleeding, our material and geometry quality suffer. A differentiable path tracer [46] can likely improve the material separation in our pipeline, but would require significantly more computations.

Our representation can be directly deployed in the vast collection of 3D content generation tools available for triangle meshes. This greatly facilitates scene editing, which is still very challenging for neural volumetric representations [70]. We show advanced scene editing examples in Figure 8, where we add our reconstructed models from the NeRFactor dataset to the Cornell box and use them in a soft-body simulation. Note that our models receive scene illumination, cast accurate shadows, and robustly act as colliders for virtual objects. In Figure 1, and the supplemental video, we show another example, where an object is reconstructed from real-world photographs and then used as a collider for a virtual cloth object. The combined scene is then rendered using our extracted environment light. Note that shading of the virtual object looks plausible given the reference photo. We also show material editing on the same example.

4.2. View interpolation

Synthetic datasets We show results for the NeRF *realistic synthetic image* dataset in Table 3 and a visual example of the MATERIALS scene in Figure 9. Per-scene results and visual examples are included in our supplemental material, where we also include Chamfer loss on extracted meshes. Our method consistently performs on par with NeRF, with better quality in some scenes. The margins are smaller for perceptually based image metrics (SSIM and LPIPS). We speculate that density-based volume approaches can more efficiently minimize PSNR than our opaque meshes. However, the effect of slightly moving a silhouette edge will not be as detrimental to a perceptual metric.

The DRUMS and SHIP scenes are failure cases for our method. We assume mostly direct lighting, with no significant global illumination effects, and these scenes contain both significant intra-scene reflections, refractions, and caustics. Interestingly, while material reconstruction suffers, we still note high quality results for view interpolation.

Method	PSNR↑	SSIM↑	LPIPS↓
PhySG	18.91	0.847	0.182
NeRF	31.00	0.947	0.081
Mip-NeRF	33.05	0.961	0.067
Our	29.05	0.939	0.081

Table 3. Average results for the eight scenes in the NeRF realistic synthetic dataset. Each scene consists of 100 training images, and 200 test images, with masks and known camera poses. Results from NeRF are taken from Table 4 of the NeRF paper [39]. PhySG and Mip-NeRF were retrained using public source code.

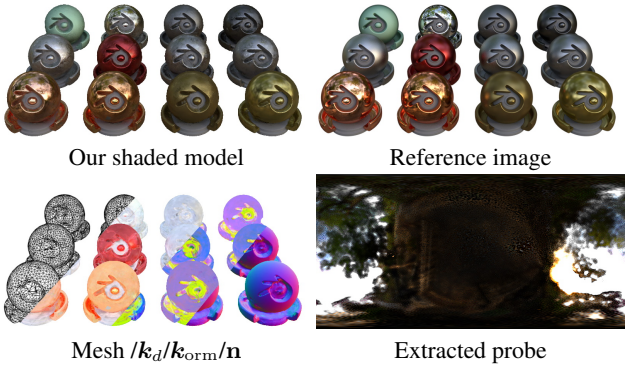


Figure 9. Our result on the MATERIALS scene, reconstructed from 100 images from the NeRF synthetic dataset.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRF	31.08	0.956	0.064
NeRFactor	26.87	0.930	0.099
Our	31.65	0.967	0.054

Table 4. View interpolation error metrics on NeRFactor's variant of the NeRF synthetic dataset. The reported image metrics are the arithmetic mean over the eight validation images of all four scenes.

Given that we factorize into explicit shape, materials and lighting, we have slightly lower quality on novel view synthesis than methods specialized for view-interpolation. To put this in context, in Table 4 we compare our aproach against NeRFactor, which performs a similar factorization, and our approach. We observe a 4.21 dB PSNR image quality reduction for NeRFactor compared to the NeRF baseline. This is consistent with NeRD [3] which do not provide source code but report a 4.17 dB quality drop for their factorized representation on a subset of the NeRF synthetic dataset. In contrast, our quality is significantly higher, while still providing the flexibility of a factorized representation.

Real-world datasets NeRD [3] provides a small dataset of real-world photo scans with auto-generated (inaccurate) coverage masks and diverse camera placement.

Visual and quantitative results are shown in Figure 10,

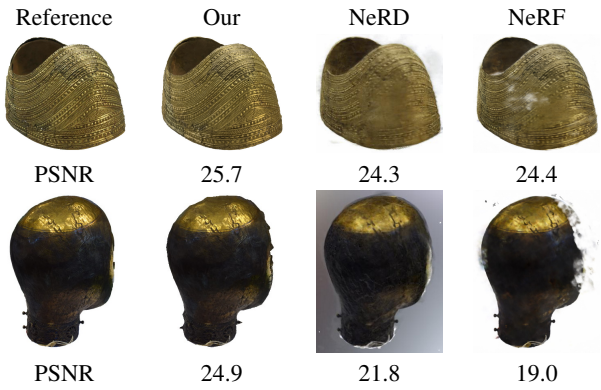


Figure 10. Reconstruction from photographs (datasets from NeRD), comparing our results with NeRD and NeRF. Images in the two rightmost colums were provided by the NeRD authors. We score higher in terms of image metrics, most likely due to our mesh representation enforcing opaque geometry, where competing algorithms rely on volumetric opacity. Despite inconsistencies in camera poses and masks, our results remain sharp while NeRF and NeRD suffer from floating or missing geometry.

where we have masked out the background for the reference object. Due to inconsistencies in the dataset, both NeRF and NeRD struggle to find sharp geometry borders with transparent "floaters" and holes. In contrast, we get sharp silhouettes and a significant boost in image quality. The results reported for NeRD are for their volumetric representation. Note that NeRD can generate output meshes as a post-processing step, but at a significant quality loss (visual comparison included in our supplemental material).

4.3. Comparing Spherical Gaussians and Split Sum

In Figure 11, we compare our differentiable split sum environment lighting approximation, from Section 3.3 against the commonly used spherical Gaussian (SG) model. Split sum captures the lighting much more faithfully across all frequencies, while still having a lower runtime cost. In our implementation, we observe a $5\times$ reduction of optimization time compared to SG with 128 lobes. At inference, evaluating the split sum approximation is extremely fast, requiring just two texture lookups.

5. Limitations and Conclusions

Our main limitation is the simplified shading model, not accounting for global illumination or shadows. This choice is intentional to accelerate optimization, but is a limiting factor for material extraction and relighting. With the current progress in differentiable path tracing, we look forward to this limitation being lifted in future work. We additionally rely on alpha masks to separate foreground from background. While our method seems quite robust to corrupted masks, it would be beneficial to further incorporate this step into the system. Other limitations include the static lighting

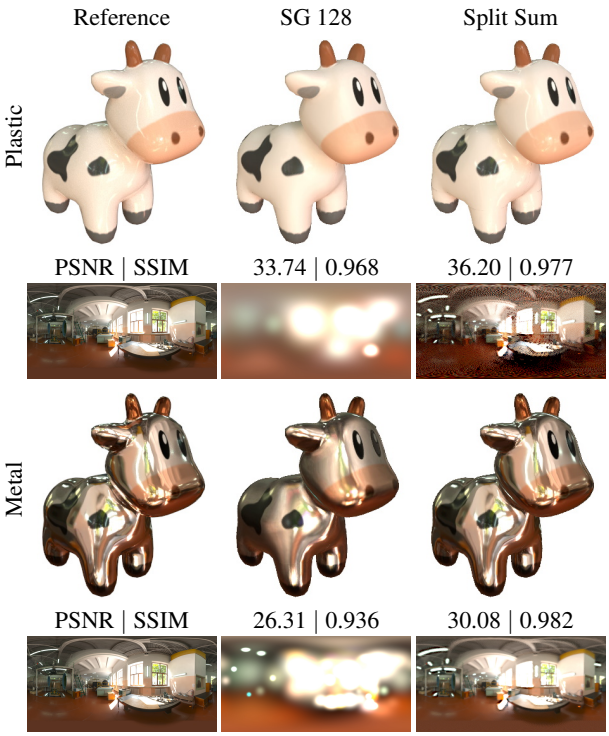


Figure 11. Environment lighting approximated with Spherical Gaussians using 128 lobes vs. Split Sum. The training set consists of 256 path traced images with Monte Carlo sampled environment lighting using a high resolution HDR probe. We assume known geometry and optimize materials and lighting using identical settings for both methods. Reported image metrics are the arithmetic mean of the 16 (novel) views in the test set. Note that the split sum approximation is able to capture high frequency lighting. Probe from Poly Haven [67].

assumption, not optimizing camera poses, and high compute resource and memory consumption during training. Apart from deepfakes, which are common to all scene reconstruction methods, we are not aware of and do not foresee nefarious use cases of our method.

In summary, we show results on par with state-of-the-art for view synthesis and material factorization, while directly optimizing an *explicit* representation: triangle meshes with materials and environment lighting. Our representation is, by design, directly compatible with modern 3D engines and modeling tools, which enables a vast array of applications and simplifies artist workflows. We perform end-to-end optimization driven by appearance of the rendered model, while previous work often sidestep the error from mesh extraction through Marching Cubes. Our method can be applied as an appearance-aware converter from a (neural) volumetric or SDF representation to triangular 3D models with materials, complementing many recent techniques.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building Rome in a Day. *Communications of the ACM*, 54(10):105–112, 2011. 2

[2] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural Reflectance Fields for Appearance Acquisition. *arXiv:2008.03824*, 2020. 2

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 5, 6, 7

[4] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-PIL: Neural Pre-Integrated Lighting for Reflectance Decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[5] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. Two-shot Spatially-varying BRDF and Shape Estimation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991, 2020. 2

[6] Brent Burley. Physically Based Shading at Disney. In *SIGGRAPH Courses: Practical Physically Based Shading in Film and Game Production*, 2012. 4

[7] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Advances In Neural Information Processing Systems (NeurIPS)*, 2019. 2, 5

[8] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khalis, Or Litany, and Sanja Fidler. DIB-R++: Learning to Predict Lighting and Material with a Hybrid Differentiable Renderer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[9] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, 1982. 5

[10] Jeremy S De Bonet and Paul Viola. Poxels: Probabilistic Voxelized Volume Reconstruction. In *Proceedings of International Conference on Computer Vision (ICCV)*, 1999. 2

[11] Yasutaka Furukawa and Jean Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 2

[12] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Gipuma: Massively Parallel Multi-view Stereo Reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V*, 25(361-369):2, 2016. 2

[13] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep Inverse Rendering for High-Resolution SVBRDF Estimation from an Arbitrary Number of Images. *ACM Transactions on Graphics*, 38(4), 2019. 2

[14] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In *Advances In Neural Information Processing Systems*, 2020. 2

[15] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-Fidelity Neural Rendering at 200FPS. *arXiv:2103.10380*, 2021. 2

[16] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear Light Source Reflectometry. *ACM Transactions on Graphics*, 22(3):749–758, 2003. 2

[17] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Estimating Specular Roughness and Anisotropy from Second Order Spherical Gradient Illumination. *Computer Graphics Forum*, 28(4):1161–1170, 2009. 2

[18] D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. BRDF Representation and Acquisition. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: State of the Art Reports*, pages 625–650, 2016. 2

[19] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. MaterialGAN: Reflectance Capture Using a Generative SVBRDF Model. *ACM Transactions on Graphics*, 39(6), 2020. 2

[20] M. Haindl and J. Filip. *Visual Texture*. Springer-Verlag, 2013. 2

[21] Andrew Svanberg Hamilton. Photogrammetry at Embark, 2021. https://medium.com/embarkstudios/photogrammetry-at-embark-part-1-88142f2e036e. 1

[22] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-Driven Automatic 3D Model Simplification. In *Eurographics Symposium on Rendering*, 2021. 2, 3, 4

[23] Krishna Murthy J., Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A PyTorch Library for Accelerating 3D Deep Learning Research. *arXiv:1911.05063*, 2019. 2

[24] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[25] James T. Kajiya. The Rendering Equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH'86*. ACM Press, 1986. 5

[26] Brian Karis. Real Shading in Unreal Engine 4. *SIGGRAPH Course: Physically Based Shading in Theory and Practice*, 2013. 3, 5, 6

[27] Michael Kazhdan and Hugues Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013. 2

[28] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015. 3

[29] Sebastien Lagarde and Charles de Rousiers. Moving Frostbite to Physically Based Rendering 3.0. *SIGGRAPH Course: Physically Based Shading in Theory and Practice*, 2014. 3

[30] 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. 2

[31] Hendrik P.A. Lensch, Jochen Lang, Asla M. Sa, and Hans-Peter Seidel. Planned Sampling of Spatially Varying BRDFs. *Computer Graphics Forum*, 2003. 2

[32] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse Rendering for Complex Indoor Scenes: Shape, Spatially-varying Lighting and SVBRDF from a Single Image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2484, 2020. 2

[33] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to Reconstruct Shape and Spatially-Varying Reflectance from a Single Image. *ACM Transactions on Graphics*, 37(6), 2018. 2

[34] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep Marching Cubes: Learning Explicit Surface Representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[35] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[36] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 4

[37] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. In *Eurographics Symposium on Rendering*, 2021. 2

[38] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7219, 2021. 2

[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, 4, 7

[40] Thomas Müller. Tiny CUDA Neural Network Framework, 2021. https://github.com/nvlabs/tiny-cuda-nn. 5

[41] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *arXiv:2201.05989*, 2022. 2, 5

[42] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11453–11464, 2021. 2

[43] 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. 2, 4

[44] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In *Eurographics Symposium on Rendering*, 2021. 2

[45] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Transactions on Graphics*, 39(4), 2020. 3

[46] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics*, 38(6), 2019. 3, 7

[47] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2, 4

[48] Marc Pollefeys and Luc Van Gool. From Images to 3D Models. *Commun. ACM*, 45(7):50–55, 2002. 1

[49] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021. 2

[50] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. *arXiv:2103.13744*, 2021. 2

[51] Carolin Schmitt, Simon Donne, Gernot Riegler, Vladlen Koltun, and Andreas Geiger. On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[52] 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. 1, 2

[53] Steven M Seitz and Charles R Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2

[54] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3, 4

[55] Smithsonian. Smithsonian 3D Digitization, 2018. https://3d.si.edu/. 4

[56] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006. 1

[57] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[58] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet Models for Refraction through Rough Surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, page 195–206, 2007. 4, 5

[59] 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. 2, 4

[60] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv:2102.07064*, 2021. 2

[61] Michael Weinmann and Reinhard Klein. Advances in Geometry and Reflectance Acquisition (Course Notes). In *SIGGRAPH Asia Courses*, 2015. 2

[62] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-time View Synthesis with Neural Basis Expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[63] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[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. 2

[65] Jonathan Young. xatlas, 2021. https://github.com/jpcy/xatlas. 5

[66] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[67] Greg Zaal. Poly Haven, 2021. https://polyhaven.com/hdris. 8

[68] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6

[69] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492*, 2020. 2

[70] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination. *ACM Transactions on Graphics*, 40(6), 2021. 1, 2, 5, 6, 7

[71] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering. In *International Conference on Learning Representations*, 2021. 2