IRON: Inverse Rendering by Optimizing Neural SDFs and Materials from Photometric Images

Kai Zhang[1] Fujun Luan[2] Zhengqi Li[3] Noah Snavely[1]

[1]Cornell University [2]Adobe Research [3]Google Research

Abstract

We propose a neural inverse rendering pipeline called IRON that operates on photometric images and outputs high-quality 3D content in the format of triangle meshes and material textures readily deployable in existing graphics pipelines. Our method adopts neural representations for geometry as signed distance fields (SDFs) and materials during optimization to enjoy their flexibility and compactness, and features a hybrid optimization scheme for neural SDFs: first, optimize using a volumetric radiance field approach to recover correct topology, then optimize further using edge-aware physics-based surface rendering for geometry refinement and disentanglement of materials and lighting. In the second stage, we also draw inspiration from mesh-based differentiable rendering, and design a novel edge sampling algorithm for neural SDFs to further improve performance. We show that our IRON achieves significantly better inverse rendering quality compared to prior works. [1]

1. Introduction

Inverse rendering—the reconstruction of shape and appearance of real-world objects from a set of 2D input images—can enable accessible, high-quality digitization of our world. One way to formulate this problem is as the inversion of rendering algorithms used in computer graphics. Recent advances in graphics have led to fully differentiable Monte Carlo path tracing methods for jointly optimizing geometry and BRDFs represented as triangle meshes and material textures. However, meshes can be difficult to optimize, because it is non-trivial to modify their topology or maintain regularity during optimization. On the other hand, recently developed neural representations for shape [22, 31] and radiance fields [23, 29] demonstrate impressive success in view synthesis [23] and shape reconstruction [26, 30, 34, 38, 39] tasks. But these representations entangle material and lighting, and cannot be directly used for applications like relighting or material editing. Recent methods for decoupling neural radiance fields are either limited to simple shapes [6, 41], or have neural components [4, 43] that are incompatible with standard 3D rendering and editing tools.

We address the inverse rendering problem with the objective of embracing both the flexibility and compactness of neural representations, and the convenience of meshes with material textures in downstream applications. We present a pipeline we call IRON—Inverse Rendering by Optimiz-

(a) Reconstruction of real-world objects, rendered under global illumination.

(b) Scene editing by modifying illumination and materials, and inserting objects.

Figure 1. Real-world objects reconstructed by our IRON pipeline. We show (a) re-renderings under novel global illumination via Mitsuba path tracing [14], and (b) scene edits that include changing lighting, material BRDFs, and the scale/orientation of existing objects, and inserting new virtual objects or participating media such as the metal Van Gogh head and the heterogeneous smoke.

[1]Our project page is at: https://kai-46.github.io/IRON-website/.

ing Neural scene components, including neural signed distance fields (SDF) and neural materials. IRON performs an inverse rendering optimization starting from multi-view images captured by co-locating a flashlight with a moving camera [4, 5, 21, 24] (called *photometric images* in recent work), while allowing for the export of the optimized 3D content in mesh and material texture formats readily usable by traditional graphics renderers and AR/VR applications, as shown in Fig. 1. At the core of IRON are four compact neural networks representing the neural SDF and materials, which we optimize with a hybrid optimization scheme: first, a volumetric radiance field optimization to recover object topology, followed by physics-based surface rendering to refine geometric details and disentangle materials and lighting. We demonstrate superior inverse rendering quality over baseline methods targeting photometric images.

In addition, recent differentiable rendering methods in the graphics community emphasize the importance of carefully considering occlusion boundaries when performing inverse rendering with meshes, e.g., with special *edge sampling* algorithms that compute accurate gradients at such edges [2, 8, 17, 18, 28, 40]. In addition to our system as a whole, we propose an edge sampling algorithm defined for neural SDF representations, rather than meshes as in prior work. We show that our new edge sampling algorithm significantly improves reconstruction quality around edges.

Contributions. To summarize, our key contributions are:

- IRON, a neural inverse rendering pipeline for high-quality reconstruction of object shape and spatially varying materials, outperforming existing methods for photometric images.
- A hybrid optimization scheme for neural SDFs and materials that first optimizes a volumetric radiance field then performs edge-aware physics-based surface rendering for improved performance and compatibility with meshes and material textures.
- An edge-aware rendering optimization featuring a novel edge sampling algorithm that generates unbiased gradient estimates for better optimizing neural SDFs.

2. Related work

Neural reconstruction and view synthesis. Neural shape [22, 31] and appearance [29] representations based on Multilayer Perceptrons (MLP) have been of recent interest, due to their compactness and representation power. Many works apply these representations to view synthesis and/or 3D reconstruction from multiple images, including NeRF [23], DVR [26], and IDR [39]. While NeRF yields view synthesis results of remarkable quality on complex scenes, its volume rendering nature leads to reduced quality of estimated surface geometry. In contrast, DVR and IDR, which utilize surface rendering, can reconstruct high-quality surface geometry, but are limited to relatively simple scenes compared to NeRF. Hence, UNISURF [30], VolSDF [38], and NeuS [34] sought to combine the benefits of both volume- and surface-based rendering by considering surfaces as defining volumes near the surface. Our work also utilizes neural shape and appearance, but rather than encoding appearance as a surface light field [36] that entangles lighting and materials, we adopt a physics-based surface shading model within an inverse rendering framework. Our pipeline can output meshes and materials readily importable by existing graphics pipelines, e.g., Blender [9], for fast raytracing, object insertion, relighting, and material editing.

Mesh-based differentiable rendering. Meshes are widely used in graphics pipelines and game engines. There has been a surge of recent interest in fully-differentiable forward rendering methods [2, 8, 17, 18, 20, 27, 28, 40], which enable joint optimization of shape, material, and camera parameters from images [21]. One challenge is proper derivative computations at depth discontinuities (called *edge derivatives*) that arise when rendering a mesh to an image. In addition, meshes are not friendly for shape optimization, due to difficulties posed in changing their topology and avoiding self-intersections [25]. In contrast, signed distance field (SDF) representations commonly used in neural geometry methods [31] parametrize the surface as the zero level set of a continuous SDF (instantiated in neural methods by an MLP), alleviating these challenges. However, we observe that, like meshes, optimizing a neural SDF via differentiable surface rendering also requires computing both interior derivatives and edge derivatives with respect to neural network weights. IDR [39] only uses interior derivatives, leading to poor performance around edges. We introduce a method to estimate edge derivatives for better optimization of neural SDFs.

Inverse rendering from multiple images. Given multiple images of a scene, inverse rendering methods seek to recover the shape, material and lighting that best explain the observed images. Prior work considers several capture scenarios, including: 1) static scene, static environmental lighting, moving camera [6, 41, 43], 2) rotating scene, static environmental lighting, fixed camera [11, 37], and 3) static scene, flashlight illumination co-located with moving camera [4, 5, 21, 24] (in some works called *photometric images*). Our work addresses the third case of flashlight photography because it simplifies the physics-based rendering module, while enabling high-quality inverse rendering results. Using neural SDFs and materials, we show improved results compared to baselines, while also creating easily deployed mesh and texture outputs.

3. Method

Assumptions. We focus on opaque objects, and consider transparent and translucent objects outside the scope of our work. We also assume that the input photometric images are

captured using collocated flashlight illumination without ambient light. Finally, for efficiency, we ignore shadows (which are minimal in practice due to the collocated flashlight) and global illumination effects like interreflections.

Under these assumptions, and provided with an input set of photometric images, our IRON system optimizes for neural shape and material representations consisting of four compact MLPs with positional encoding [23, 32]:

Neural SDF $S_{\boldsymbol{\Theta}_s} : \boldsymbol{x} \rightarrow (S, \boldsymbol{f})$ maps a 3D location $\boldsymbol{x}$ to an SDF value S and a 256D local geometric feature descriptor $\boldsymbol{f}$, as in recent works [30, 34, 39]; this feature descriptor can then be fed into our neural material networks.

Neural diffuse albedo $\beta_{\boldsymbol{\Theta}_\beta} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \beta$ outputs the diffuse albedo β at location $\boldsymbol{x}$ given surface normal $\boldsymbol{n}$ and feature descriptor $\boldsymbol{f}$. This function plays a dual role in our optimization. In our first optimization phase, we treat β as a neural radiance field, and include view direction as an additional parameter (in place of the second $\boldsymbol{n}$). In the second phase, we constrain β to represent diffuse albedo in our edge-aware physics-based surface rendering. Our neural diffuse albedo MLP utilizes the surface normal and local geometric feature descriptors as in prior work [30, 34, 39].

Neural specular albedo $\kappa_{\boldsymbol{\Theta}_\kappa} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \kappa$ encodes spatially-varying specular albedo κ.

Neural roughness $\alpha_{\boldsymbol{\Theta}_\alpha} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \alpha$ encodes the spatially-varying specular roughness. Small values indicate shiny surfaces, and large values less shiny.

We optimize the MLP weights $\boldsymbol{\Theta}_s, \boldsymbol{\Theta}_\beta, \boldsymbol{\Theta}_\kappa$, and $\boldsymbol{\Theta}_\alpha$ and a scalar light intensity L in a two-stage optimization scheme. In the first stage, we optimize neural SDF $S_{\boldsymbol{\Theta}_s}$ and diffuse albedo $\beta_{\boldsymbol{\Theta}_\beta}$ by treating $\beta_{\boldsymbol{\Theta}_\beta}$ as a volumetric radiance field. This phase is designed to recover correct object topology and serves as an initialization for the second phase. In the second phase, we refine geometric details and factorize materials from lighting by jointly optimizing neural SDF $S_{\boldsymbol{\Theta}_s}$, neural materials $\beta_{\boldsymbol{\Theta}_\beta}, \kappa_{\boldsymbol{\Theta}_\kappa}$, and $\alpha_{\boldsymbol{\Theta}_\alpha}$, and light intensity L via an edge-aware physics-based surface rendering method.

3.1. Volumetric radiance field rendering

In the first stage, we optimize $\beta_{\boldsymbol{\Theta}_\beta}(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{n}, \boldsymbol{f})$ as a view-dependent neural radiance field (by substituting view direction $-\boldsymbol{d}$ for the second $\boldsymbol{n}$). Hence, we perform volumetric radiance field rendering of the neural SDF $S_{\boldsymbol{\Theta}_s}$ and colors $\beta_{\boldsymbol{\Theta}_\beta}(\boldsymbol{x}, \boldsymbol{n}, -\boldsymbol{d}, \boldsymbol{f})$ as in [34] in order to harness the power of volume rendering in recovering correct object topology [23, 30, 34, 38], i.e., the correct number and location of holes in the geometry. If we do not perform this initial optimization stage and instead optimize for surface rendering from scratch, we found that the optimization frequently diverges unless object segmentation masks are provided, as observed by [30], and often gets stuck in local minima with incorrect topology. At the same time, the volumetric nature of this stage is inconsistent with our goal of producing shape and materials that are compatible with the mesh-based rendering paradigm used in existing graphics pipelines. This motivates the second optimization stage where we perform edge-aware physics-based surface rendering.

Algorithm 1 Locate edge points

Input: ray-surface intersection $\hat{\boldsymbol{x}}$
Output: an edge point or NOT_FOUND.
Hyperparams: max # steps K, step size ϵ, threshold δ.

$\boldsymbol{x}_t \leftarrow \hat{\boldsymbol{x}}$
for $i \leftarrow 1$ to K **do**
  if $(\frac{\boldsymbol{x}_t - \boldsymbol{o}}{\|\boldsymbol{x}_t - \boldsymbol{o}\|_2})^T \boldsymbol{n}_t < \delta$ **then**
    return $\boldsymbol{x}_t$
  else
    $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t + \epsilon \cdot (\boldsymbol{n}_t - \frac{\boldsymbol{o} - \boldsymbol{x}_t}{(\boldsymbol{o} - \boldsymbol{x}_t)^T \boldsymbol{n}_t})$
  end if
end for
return NOT_FOUND

3.2. Edge-aware physics-based surface rendering

After solving for a volumetric radiance field, we perform a second, full inverse rendering stage that jointly optimizes neural SDF $S_{\boldsymbol{\Theta}_s}$ and neural materials $\beta_{\boldsymbol{\Theta}_\beta}, \kappa_{\boldsymbol{\Theta}_\kappa}$, and $\alpha_{\boldsymbol{\Theta}_\alpha}$) from photometric images. This stage has two key components: differentiable physics-based shading and edge-aware surface rendering.

Physics-based shading. As our photometric image inputs have co-located flashlight and camera, the light direction is aligned with the view direction across pixel locations. Hence, we can simplify the rendering equation [16] as:

$$L_o(\boldsymbol{\omega}_o, \mathbf{x}) = \int_\Omega L_i(\boldsymbol{\omega}_i, \mathbf{x}) f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i, \mathbf{x})(\boldsymbol{\omega}_i \cdot \mathbf{n}) d\boldsymbol{\omega}_i \quad (1)$$

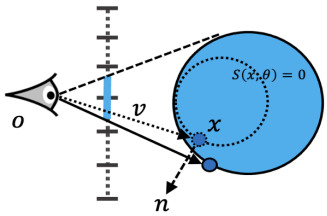
$$\approx L_i(\boldsymbol{\omega}_o, \mathbf{x}) f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_o, \mathbf{x})(\boldsymbol{\omega}_o \cdot \mathbf{n}), \quad (2)$$

where $L_o, \mathbf{x}, \mathbf{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, L_i, f_r$ are observed light, surface location, surface normal, light direction, view direction, incident light and BRDF, respectively. We model the white flashlight as a point light source as in [21, 24]:

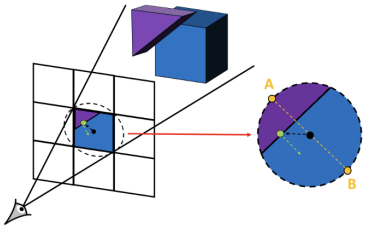
$$L_i(\boldsymbol{\omega}_o; \mathbf{x}) = \frac{L}{\|\mathbf{x} - \boldsymbol{o}\|_2^2}, \quad (3)$$

where L is a scalar light intensity and $\boldsymbol{o}$ is the light location (same as the camera location in our co-located capture setup). The denominator models the inverse-square fall-off in intensity. For BRDF f_r, we use the GGX model [33], whose parameters at location $\boldsymbol{x}$, diffuse albedo β, specular albedo κ and roughness α, are encoded in our neural materials $\beta_{\boldsymbol{\Theta}_\beta}, \kappa_{\boldsymbol{\Theta}_\kappa}$, and $\alpha_{\boldsymbol{\Theta}_\alpha}$, respectively.

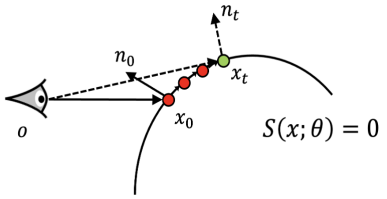
From Eqns. 2 and 3, we observe that gradients from the rendered image $L_o(\boldsymbol{\omega}_o; \mathbf{x})$ must back-propagate through sur-



(a) Assuming known color, prior neural surface rendering methods [26, 39] fail to deform a small sphere to a target large sphere due to lack of edge handling.



(b) The shading at an edge pixel involves the combination of the shading at disconnected surface pieces identified using subpixel localization of edges.



(c) We locate edge points in 3D by walking on the zero level set of the neural SDF, then project 3D edge points into 2D for subpixel edge localization.

Figure 2. Illustration of edge-aware surface rendering for neural SDFs. (a) Existing neural surface rendering methods ignore geometric discontinuities, making it difficult to deform neural SDFs to match silhouettes even for simple objects. (b) Geometric discontinuities are introduced by edge pixels where multiple depth values are present in a single pixel, motivating our proposed edge sampling algorithm. (c) Our method can localize subpixel-accurate edges for neural SDFs enabling correct shading calculations at edge pixels.

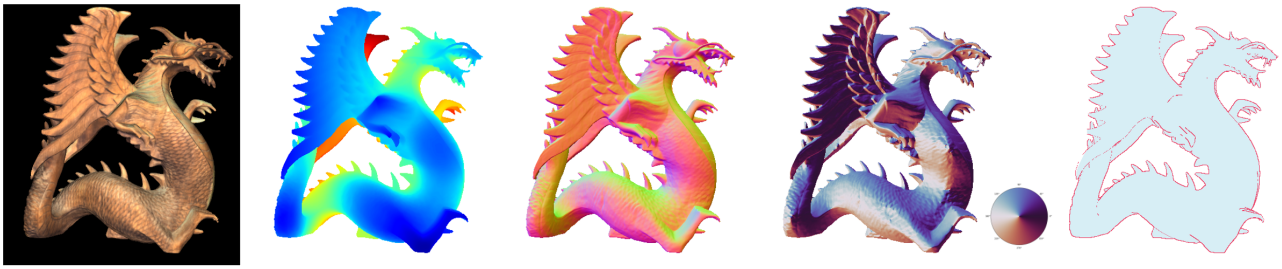


Figure 3. Demonstration of edge-aware surface rendering for neural SDF. To render the image (a), we first run sphere tracing to get the depth (b) and surface normal (c) corresponding to the pixel centers. We then localize subpixel-accurate edges by walking on the surface; the projected walk direction (d) is color-coded as shown in the inset color wheel. Detected edge pixels are shown in red in (e). We shade non-edge pixels as in existing neural surface rendering methods [26, 39], but shade edge pixels by accounting for the geometric discontinuity.

face location $\boldsymbol{x}$ and surface normal $\boldsymbol{n}$ to the shape parameters, and through the BRDF f_r to the material parameters.

Edge-aware surface rendering. Inspired by recent advances in mesh-based differentiable rendering [2, 18, 21, 28, 40], we identify a key issue in prior neural surface rendering works such as IDR [39] and DVR [26]. Namely, their differentiable rendering module only works for interior pixels, due to their assumption of a smooth surface inside each pixel footprint. This assumption fails for edge pixels, where shading color is a combination of colors at disconnected surface pieces, as shown in Fig. 2(b). For this reason, these methods compute biased gradients w.r.t. the weights of the neural SDF that move surface points along camera rays, but not in the image plane, due to missing edge gradients reflecting how color changes w.r.t. edge location. In Fig. 2(a), we illustrate how these missing edge gradients lead to non-convergence in the simple task of deforming an initial small sphere to a target large sphere assuming known shading color. This problem is confirmed empirically in Fig. 4.

We address this issue via a novel edge sampling algorithm tailored for neural SDFs. Our algorithm has three steps: 1) localize subpixel-accurate 2D edges by detecting 3D edge points that are then projected to the 2D image, 2) reparametrize edge points such that they can back-propagate gradients to the neural SDF in an auto-differentation framework, and 3) compute the shading color for edge pixels.

In step 1, we start from ray-surface intersections found by sphere tracing the center rays at each pixel location [13] (Fig. 3(b,c)), and walk on the surface along the direction defined in line 6 of Alg. 1 and illustrated in Fig. 2(c) until reaching a 3D edge point or a max number of steps [3, 7]. We visualize projected walk directions in Fig. 3(d). For the sake of efficiency, we only do the surface walk process for the ray-surface intersections at depth discontinuity pixels in order to reduce the number of evaluations of the neural SDF; we identify such depth discontinuity pixels as ones with depth Sobel gradient magnitude above a certain threshold τ. We then project detected 3D edge points to image space, producing both subpixel-accurate edge locations and an edge mask marking pixels containing edges (Fig. 3(e)). We also obtain 2D edge normal directions by projecting the 3D surface normals at these edge points to 2D.

In step 2, we reparameterize the edge points' locations $\boldsymbol{x}$ to make them differentiable with respect to network weights

of the neural SDF. We observe that the differentiable ray-surface intersection equation (Eq. 4) for interior points in [26, 39] only captures how perturbations to neural SDF weights move the ray-surface intersection along the camera ray, while for edge points, we care about their movement along the surface normal. Hence, to reparameterize edge points correctly, we propose to replace the viewing direction $\boldsymbol{o} - \boldsymbol{x}$ ($\boldsymbol{o}$ is the camera origin, $\boldsymbol{x}$ is a surface point) in Eq. 4 with the surface normal $\boldsymbol{n}$, and arrive at Eq. 5.

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} - \frac{\boldsymbol{o} - \boldsymbol{x}}{\boldsymbol{n}^T(\boldsymbol{o} - \boldsymbol{x})} S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}), \tag{4}$$

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} - \frac{\boldsymbol{n}}{\boldsymbol{n}^T\boldsymbol{n}} S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{n} S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}). \tag{5}$$

We show the correctness of our edge point reparametrization in the supplemental material.

In step 3, we compute the shading at each edge pixel. Consider the edge pixel in Fig. 2(b), and let the green projected edge point have subpixel coordinates $[u, v]$, and the green projected surface normal be $[du, dv]$. Then the black center of this edge pixel has coordinates:

$$[u_c, v_c] = \big[\text{floor}(u), \text{floor}(v)\big] + 0.5. \tag{6}$$

We approximate each square pixel footprint using a circle of radius $\frac{\sqrt{2}}{2}$ pixels centered at $[u_c, v_c]$. We then pick 2D locations, labeled A and B, on the circle on either side of the edge. We raytrace and shade the two selected 2D locations. Let us denote the shaded colors as C_A and C_B, respectively. We linearly combine C_A and C_B with weights proportional to the two segment areas separated by the edge. Suppose the fraction of segment area on the same side as A is $w_A \in [0, 1]$:

$$\alpha = 2 \cdot \arccos(\sqrt{2} \cdot [du, dv][u - u_c, v - v_c]^T), \tag{7}$$

$$w_A = 1 - \frac{1}{2\pi} \cdot \big(\alpha - \sin\alpha\big), \tag{8}$$

then our predicted edge pixel color is:

$$C = w_A C_A + (1 - w_A) C_B. \tag{9}$$

In Eq. 9, gradients from C can back-propagate through w_A, C_A, and C_B to the parameters of our neural SDF and materials, and through the variable w_A, we correctly model how tiny perturbations of the 3D edge point's location along the surface normal direction affect the edge pixel color.

3.3. Training and testing

Given multi-view photometric images, we optimize our neural SDF and materials using the following loss:

$$\begin{aligned} L = & L_2\big(\text{pyramid}(\hat{I}), \text{pyramid}(I)\big) && (10)\\ & + 1 - \text{SSIM}(\hat{I}, I) && (11)\\ & + \lambda_1 \cdot \|\nabla_x S - 1\|_2^2 && (12)\\ & + \lambda_2 \cdot \max(\text{roughness}(x) - 0.5, 0), && (13) \end{aligned}$$

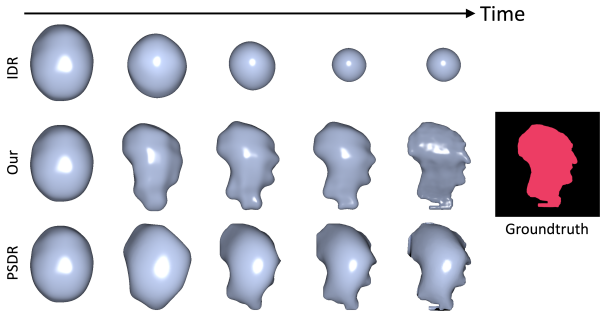


Figure 4. Given a single target image of an object with known color, we use an image loss to optimize a neural SDF with both IDR (top row) [39] and out IRON (middle row) method. We also optimize a mesh using PSDR (bottom row) [21]. Results show that IDR gets stuck due to not allowing SDF to deform along the image plane, while our method correctly optimizes the sihouette. PSDR can also handle the silhouettes, but the mesh quality degrades without intermediate remeshing steps.

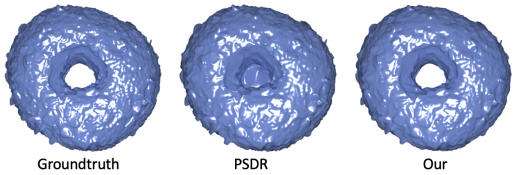


Figure 5. When using a mesh representation, it is difficult to design methods that can change the mesh topology in a differentiable way. In contrast, methods that use neural SDFs can more readily change their topology. Above, we use the mesh-based differentiable rendering pipeline PSDR [40] and our IRON pipeline to reconstruct this bagel by deforming an initial sphere. The results demonstrate that PSDR fails to recover the central hole, while our method has no such issue.

where Eq. 10 is the L_2 loss on Gaussian pyramids of the predicted image $\hat{I}$ and groundtruth image I, Eq. 11 is the SSIM loss [35], Eq. 12 is the eikonal loss [10, 12] enforcing the validity of the SDF, and Eq. 13 is the roughness range loss encouraging the estimated roughness to stay below 0.5. λ_1, λ_2 are loss weights.

Once training is complete, we convert the neural SDF and materials to a triangle mesh and texture map for deployment in the standard graphics pipeline. We first extract a mesh from the optimized neural SDF using the marching cube algorithm [19]. We then use the Blender Smart UV Project tool [9] to compute a reasonable per-vertex uv mapping. Finally, to fill the material texture images, we densely sample points on our triangle meshes with trilinearly interpolated uv coordinates, then query the material networks at the sampled surface points, and finally splat the per-point material parameters to the texture images using their interpolated uv coordinates.

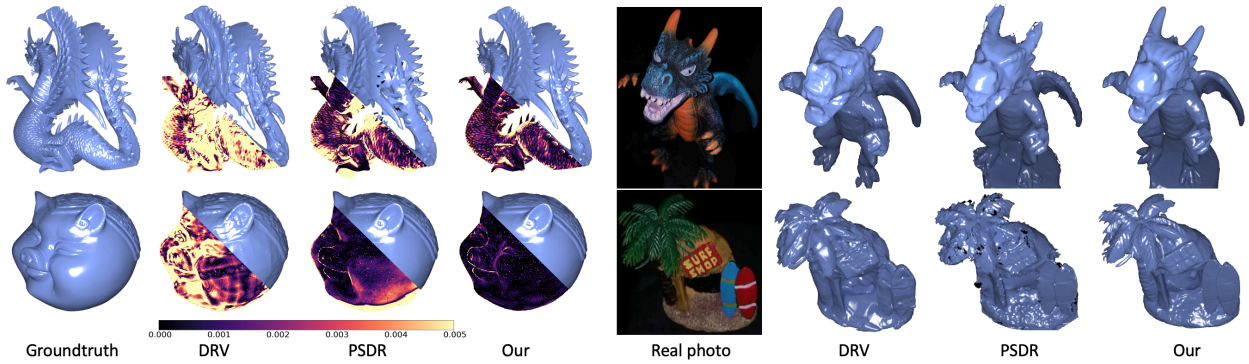


Figure 6. Comparison of recovered geometry on synthetic (**left**) and real data (**right**) with baselines DRV [4] and PSDR [21]. On synthetic data, we achieve the highest geometric reconstruction accuracy among the inverse rendering methods designed for flashlight photography. For real data, we show a captured photo for reference due to lack of ground truth geometry.

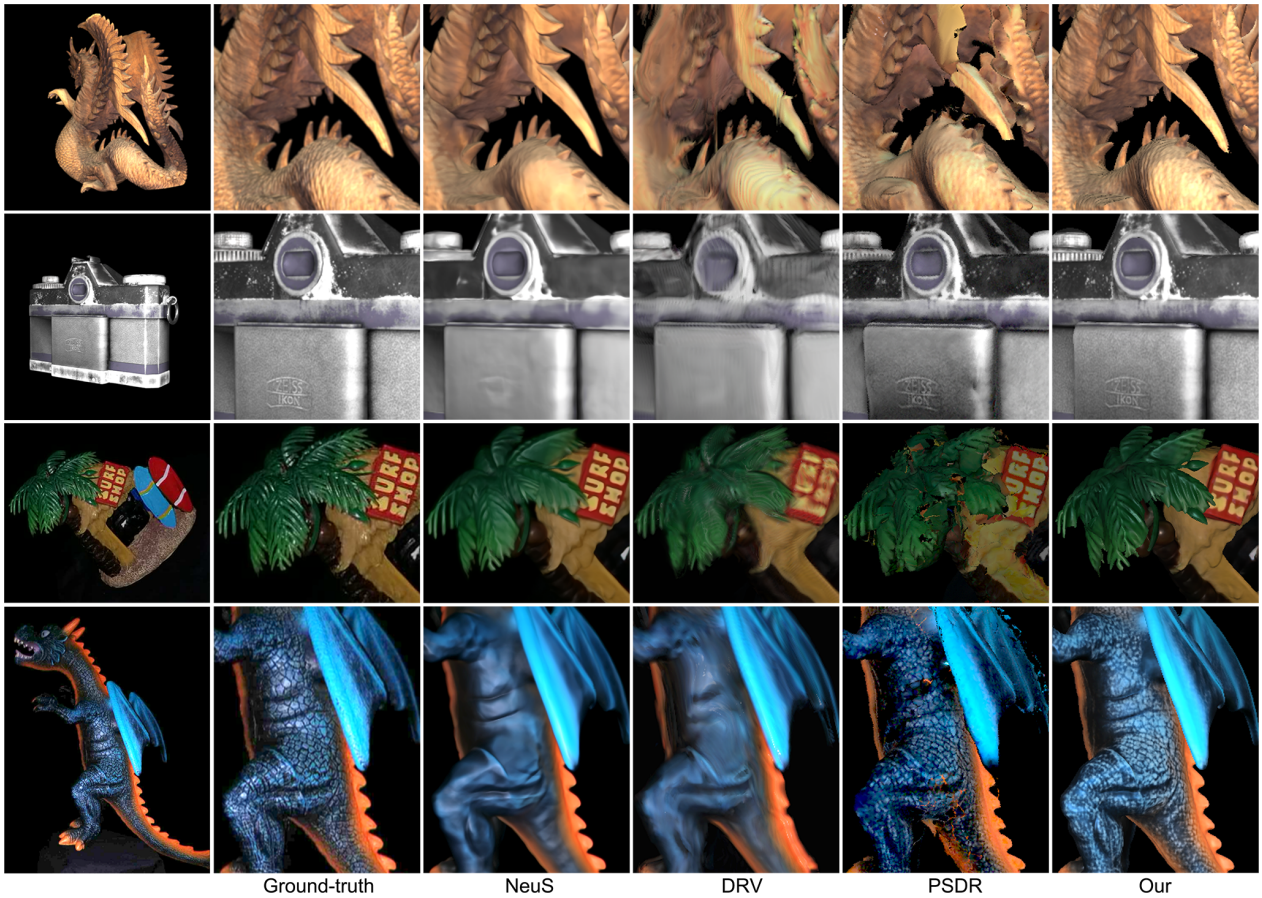


Figure 7. Qualitative comparison of generalization to novel co-located flashlight relighting using both synthetic (**top two rows**) and real (**bottom two rows**) data. Our IRON method produces visually more faithful matches in terms of geometric and texture details compared to DRV [4] and PSDR [21]. Results of the recent neural surface reconstruction work NeuS [34] are also included for reference.

4. Evaluation

We perform extensive experiments to validate IRON.

4.1. Optimizing neural SDFs to fit single image

We design a controlled experiment to illustrate the key effect of our edge sampling algorithm: allowing the image loss to lead to in-plane neural SDF deformations. In this experiment, we assume that a single image of a 3D object with

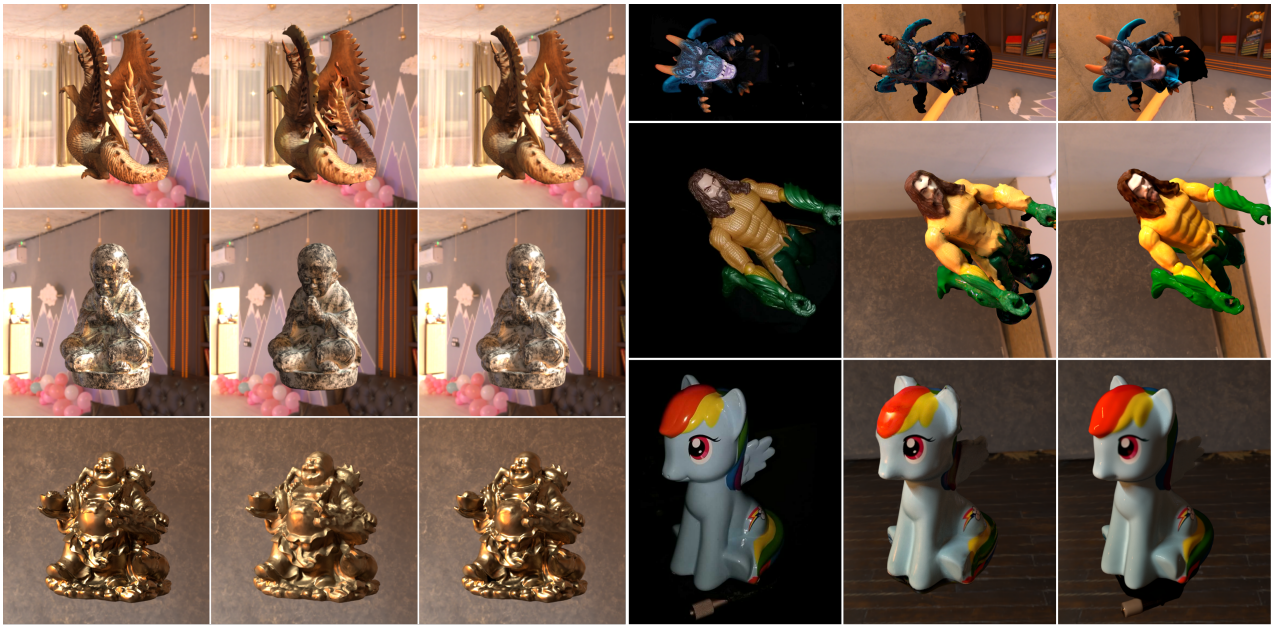


Figure 8. Comparison of generalization to novel natural environmental lighting on both synthetic (**left**) and real (**right**) data. We render both the PSDR [21] and our reconstructions under environmental illuminations using Mitsuba [14]. Note that our IRON pipeline matches the ground truth better for specular highlight regions on synthetic data, and is more perceptually convincing on real data.

	↓Chamfer L1 [1]	↓LPIPS [42]	↑SSIM [35]	↑PSNR
DRV [4]	0.0111	0.1133	0.8252	28.0693
PSDR [21]	0.0048	0.1032	0.9358	27.1354
Our	**0.0014**	**0.0438**	**0.9747**	**31.2614**

Table 1. Quantitative comparisons with baseline methods on synthetic data in terms of geometry quality and relighting quality under novel co-located flashlight illumination. Note all synthetic objects are scaled to lie inside the unit sphere before evaluation.

	↓LPIPS [42]	↑SSIM [35]	↑PSNR
DRV [4]	**0.1016**	0.8264	**32.0303**
PSDR [21]	0.1861	0.8137	25.7452
Our	0.1091	**0.8614**	29.3694

Table 2. Quantitative comparisons on real data in terms of relighting quality under novel co-located flashlight illumination.

known constant color is given as input to optimize a neural SDF through an image loss. To solve this problem, the neural SDF must be deformed such that its silhouette matches the ground truth when viewed from the input viewpoint. But as shown in Fig. 4, the prior neural surface rendering method IDR [39] fails to accomplish this task due to a lack of edge pixel handling. Our edge-aware surface rendering method addresses this problem, converging to a valid solution. Although mesh-based differentiable rendering methods can also handle silhouettes [40], we observe degraded mesh quality over the course of optimization without intermediate remeshing, due to the otherwise fixed mesh topology. In contrast, neural SDFs do not suffer from such problems.

4.2. Inverse rendering from photometric images

We now evaluate methods on the task of inverse rendering from multi-view photometric images.

Datasets. We create a synthetic dataset consisting of 9 objects: *dragon*, *buddha*, *camera*, *monk*, *kettle*, *duck*, *pig*, *sneaker*, and *bagel*. Each object is rendered from 200 randomly sampled viewpoints using the Mitsuba path-tracing renderer [14] to form the training data. We co-locate a point light source with the camera to actively illuminate the object without other light sources. We render test data consisting of 100 images under novel co-located flashlight illumination, and another 100 images under novel natural environmental illumination. For real-world data, we use 5 object captures from DRV [5]: *dragon*, *pony*, *girl*, *tree*, and *triton*. The data is acquired using co-located flashlight setup in a dark environment. We randomly choose 70% of the images for training and use the remaining 30% as test images for evaluating generalization to novel view with co-located lighting.

Baselines. We compare with two inverse rendering methods that use densely captured photometric images: the volume-based DRV [5] and the mesh-based PSDR [21] methods. We did not compare with the mesh-based method of Nam *et al*. [24] due to the lack of open-source code. In addition, PSDR has shown superior reconstruction quality compared to this method.

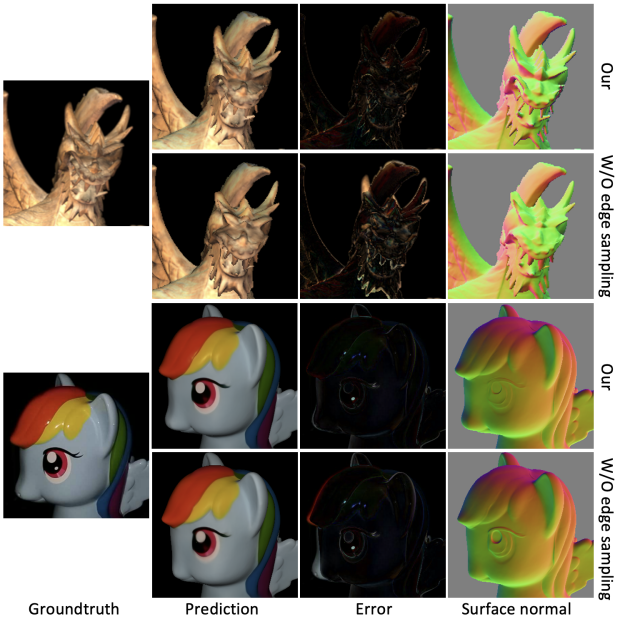


Figure 9. Ablations on our edge sampling algorithm using the synthetic dragon and real pony data. Without edge sampling, neural surface rendering performs poorly around edge regions, as shown by the error maps and surface normals.

Discussion. As shown in Tab. 1 and Figs. 6 and 7, we outperform state-of-the-art inverse rendering baselines by a large margin in terms of geometric accuracy and generalization to novel co-located lighting on synthetic data, and produce fewer artifacts like blurry textures on real data. The mesh-based PSDR method sometimes produces geometry with incorrect topology (number of holes), as shown by the bagel data in Fig. 5. Such mesh-based methods also face difficulties in maintaining mesh regularity and avoiding self-intersections during optimization, and require repeated application of error-prone remeshing and uv-mapping procedures. Neural SDFs avoid these problems with their continuous representation. We also compare with PSDR in terms of generalization to novel environmental lighting in Fig. 8. IRON results in better synthesized specular highlights and more perceptually convincing relighting. The volumetric DRV results are not directly renderable by Mitsuba; hence we did not compare with them under novel environmental lighting. On real data under novel co-located flashlight relighting, IRON's material estimates are much more detailed than those of baseline methods (Fig. 7), with quantitative metrics on par with DRV (Tab. 2). While DRV has slightly better LPIPS and PSNR scores, we observe that it significantly blurs detail. In addition, our method enjoys the advantage of simple conversion to a mesh for computer graphics deployment.

Ablations on edge sampling. We test removing the our edge sampling method from IRON. As shown in Fig. 9, the resulting reconstructions have much poorer quality around edge regions, e.g., the dragon horn and pony nose. This poor quality is due to the missing edge derivatives in existing neural surface rendering methods [26, 39].

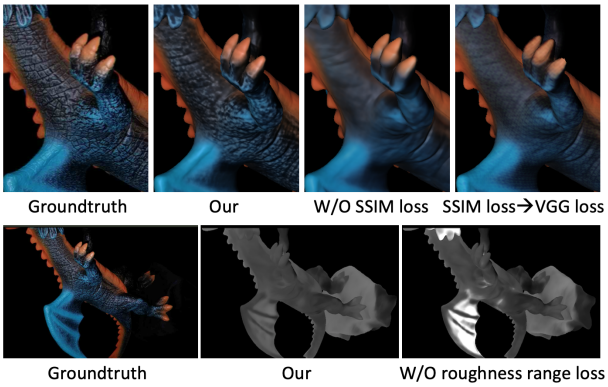


Figure 10. Ablations on our loss functions using the real dragon data. Without patch-based SSIM loss, our synthesized image is quite blurry, while VGG loss results in spurious texture details. The roughness range loss fixes issue of over-estimating roughness.

Ablations on loss functions. We find that using the SSIM loss significantly improves the sharpness of reconstructions. On the real dragon data, we first try removing SSIM loss from our pipeline. Then we also try replacing SSIM loss with VGG loss [15], which is widely used in view synthesis tasks. As shown in Fig. 10, our proposed loss functions lead to the sharpest and most plausible material reconstructions.

5. Conclusion

In this work, we have presented our new IRON pipeline specialized in high-quality inverse rendering from photometric images. In constrast to mesh-based differentible rendering, IRON adopts neural SDFs and materials as representations for ease of optimization in the form of both volumetric radiance rendering and edge-aware physics-based surface rendering, and preserves the convertibility to meshes and material textures for the sake of downstream applications.

Limitations and future work. There are limitations to be resolved in future work. First, the use of photometric images leads to a more involved data capture process, although such images simplify inverse methods because of the known single point light and minimal shadows. Future work can explore extensions of our work to the combination of flashlight and ambient illumination. Second, we do not model multiple bounces of light. This can lead to material estimation errors in concave regions that feature significant interreflection. Future directions include devising computationally-efficient global illumination rendering algorithms for neural SDF representations. Third, our current BRDF model assumes opaque surfaces, and thus we do not expect our method to work well on transparent and translucent objects that feature

significant refraction and subsurface scattering.

Acknowledgements. This work was supported in part by the National Science Foundation (IIS-2008313), and by funding from Intel and Amazon Web Services. We also thank Sai Bi for providing their code and data.

A. Neural network structures

Neural SDF $S_{\boldsymbol{\Theta}_s} : \boldsymbol{x} \rightarrow (S, \boldsymbol{f})$. We use an 8-layer MLP of width 256 and a skip connection at the 4th layer. The input 3D location $\boldsymbol{x}$ is encoded by positional encoding using 6 frequencies to compensate the spectral bias of MLPs [23, 32].

Neural diffuse albedo $\beta_{\boldsymbol{\Theta}_\beta} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \beta$. We use a 8-layer MLP of width 256 and a skip connection at the 4th layer. The input 3D location $\boldsymbol{x}$ is positional-encoded using 10 frequencies. The second surface normal $\boldsymbol{n}$ (equals the viewing direction in the first stage of volumetric radiance fields rendering) is positional-encoded using 4 frequencies.

Neural specular albedo $\kappa_{\boldsymbol{\Theta}_\kappa} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \kappa$. We use a 4-layer MLP of width 256, with input 3D location $\boldsymbol{x}$ positional-encoded using 6 frequencies.

Neural roughness $\alpha_{\boldsymbol{\Theta}_\alpha} : (\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{f}) \rightarrow \alpha$. We use a 4-layer MLP of width 256, with input 3D location $\boldsymbol{x}$ positional-encoded using 6 frequencies.

B. Implementation details

We implement our BRDF by following the Mitsuba roughplastic BRDF implementation [14] closely. The *distribution* parameter in Mitsuba roughplastic BRDF is chosen as "ggx", while the *intIOR*, *extIOR*, and *nonlinear* parameters are set to their default values.

During the volumetric radiance field rendering optimization stage, we train for $100k$ iterations using 512 randomly sampled pixels at each iteration with ℓ_1 image loss and the eikonal regularization loss (weight $\lambda_1 = 0.1$). During the edge-aware physics-based surface rendering stage, we set the eikonal loss weight $\lambda_1 = 0.1$, and the roughness range loss weight $\lambda_2 = 0.1$. We set $\tau = 1\text{e}{-2}$ for the depth gradient magnitude threshold, $K = 16$ for the maximum number of surface walk steps, $\epsilon = 1\text{e}{-3}$ for the step size, and $\delta = 5\text{e}{-2}$ for the dot-product threshold when localizing edge points. At each training iteration, we render a random 128×128 image patch to compare with ground truth. The number of Gaussian pyramid levels for ℓ_2 image loss is set to 4. We train the second stage for $50k$ iterations. The two stages take $\sim$10 hours on a single NVIDIA RTX2080Ti GPU with 12G memory.

C. Proof of edge point re-parametrization

In this section, we prove the correctness of our edge point re-parametrization in Eq. (4) of the main paper. Our proof strategy is similar to the differentiable ray-surface intersection in [26, 39], and divided into 3 steps: 1) re-parametrize the edge point in terms of the target moving direction, then show both 2) the function value and 3) the first derivative value under current parameter setting.

First, we note that our goal is to move a 3D edge point $\boldsymbol{x}$ along its surface normal direction $\boldsymbol{n}$; hence:

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} + t_{\boldsymbol{\Theta}_s} \cdot \boldsymbol{n}. \tag{14}$$

Second, we note that under current parameter setting, i.e., $\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}$, we have:

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = \boldsymbol{x}. \tag{15}$$

This is to say that the function $\boldsymbol{x}_{\boldsymbol{\Theta}_s}$ evaluated at $\boldsymbol{\Theta}_s^{(0)}$ equals the edge point location $\boldsymbol{x}$. Substituting Eq. 14 into Eq. 15 gives:

$$t_{\boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = 0. \tag{16}$$

Third, we note that $\boldsymbol{x}_{\boldsymbol{\Theta}_s}$ must stay on the zero level set during deformation:

$$S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}_{\boldsymbol{\Theta}_s}) = 0. \tag{17}$$

Implicit-differentiate with respect to $\boldsymbol{\Theta}_s$ on both sides, and we have:

$$\frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x}_{\boldsymbol{\Theta}_s}) + \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)^T \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\Theta}_s} = 0. \tag{18}$$

Differentiating both sides of Eq. 14 with respect to $\boldsymbol{\Theta}_s$ gives:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\Theta}_s} = \boldsymbol{n} \frac{\partial t}{\partial \boldsymbol{\Theta}_s}. \tag{19}$$

Substituting Eq. 19 into Eq. 18 gives us:

$$\frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x}_{\boldsymbol{\Theta}_s}) + \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)^T \boldsymbol{n} \frac{\partial t}{\partial \boldsymbol{\Theta}_s} = 0. \tag{20}$$

Evaluating Eq. 20 at $\boldsymbol{\Theta}_s^{(0)}$, substituting $\boldsymbol{n} = \frac{\partial S}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}}$ and Eq. 15, we have:

$$\frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x})\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} + \boldsymbol{n}^T \boldsymbol{n} \frac{\partial t}{\partial \boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = 0. \tag{21}$$

Rearranging a bit, we have:

$$\frac{\partial t}{\partial \boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = -\frac{1}{\boldsymbol{n}^T \boldsymbol{n}} \cdot \frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x})\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}}. \tag{22}$$

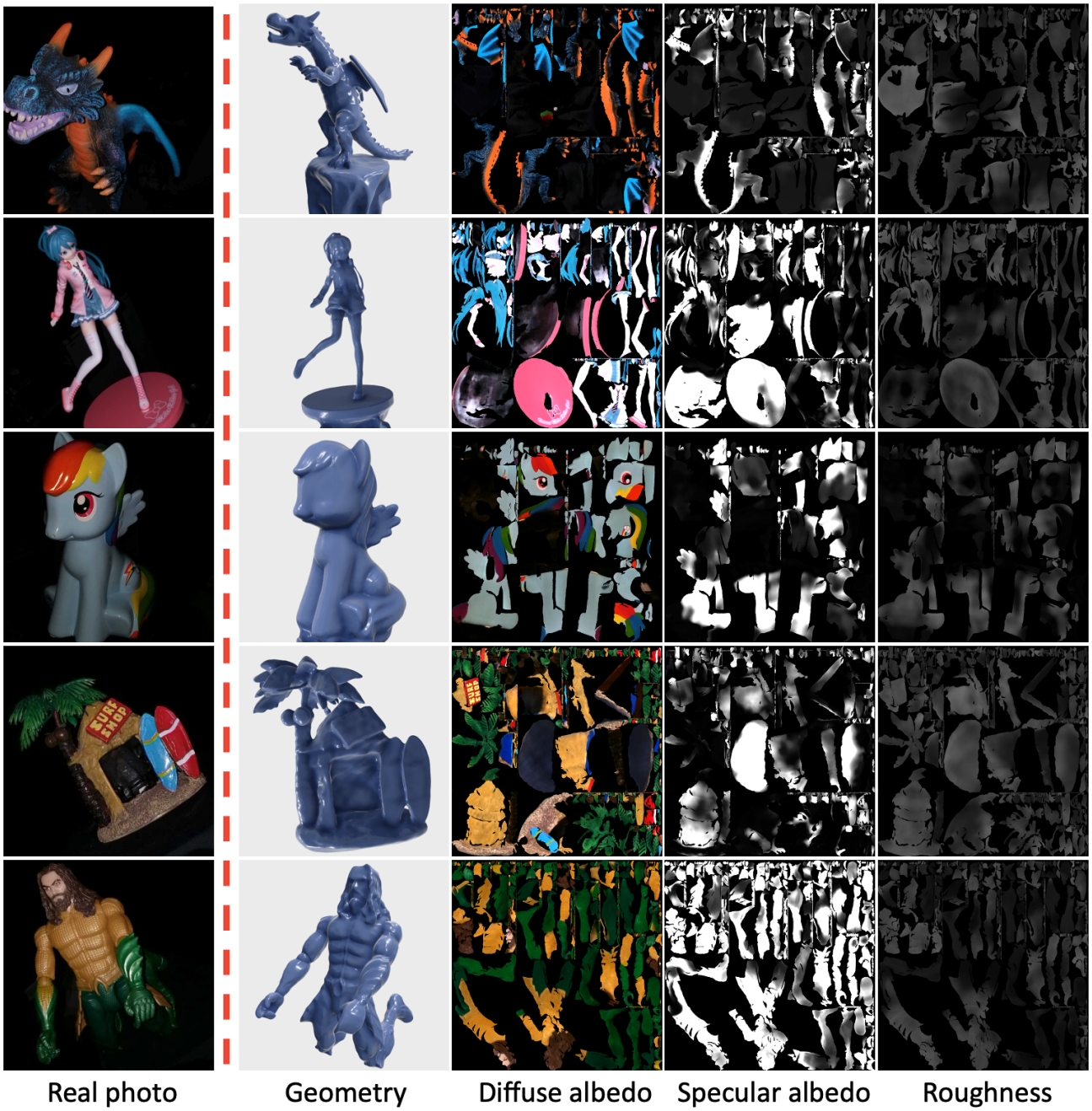


Figure 11. Reconstructed meshes and materials for the real scenes by our IRON system.

Finally, we conclude our proof by observing that Eqns. 16,22 gives us the first order approximation of $t_{\boldsymbol{\Theta}_s}$:

$$t_{\boldsymbol{\Theta}_s} = -\frac{1}{\boldsymbol{n}^T\boldsymbol{n}} \cdot S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}). \tag{23}$$

Substituting Eq. 23 into Eq. 14 leads to our proposed edge point re-parametrization:

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} - \frac{\boldsymbol{n}}{\boldsymbol{n}^T\boldsymbol{n}} \cdot S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}). \tag{24}$$

We note that $\boldsymbol{x}_{\boldsymbol{\Theta}_s}$ only provides unbiased first-order gradient with respect to $\boldsymbol{\Theta}_s$ suitable for gradient-based optimizers.

D. Reconstructed meshes and materials

In Fig. 11, we show the reconstructed meshes and materials for the 5 real-world scenes used in this work.

E. Comparison with PhySG

First, note there are a few key limitations to PhySG that our method can overcome. Namely, unlike our method, PhySG requires input object segmentation masks. Our method can also handle spatially-varying specular roughness, whereas PhySG assumes a constant and uniform specular lobe shape. Because PhySG assumes static environmental lighting, we used Mitsuba to render the same object from the same set of viewpoints first using a collocated flash (to run our method), and then with environmental lighting (to run PhySG). See Fig. 12 and the caption for an example comparison.

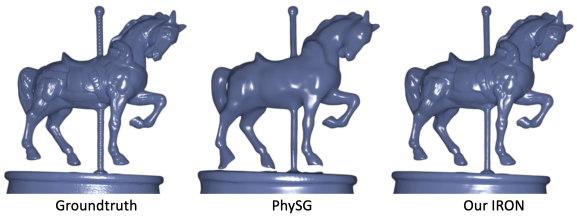


Figure 12. For the horse object, our method recovers much more accurate geometry details than PhySG; chamfer L1 distances are: Our IRON (5.35e-4) vs. PhySG (18.67e-4).

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *Int. J. Comput. Vis.*, pages 1–16, 2016. 7

[2] Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020. 2, 4

[3] Pierre Bénard and Aaron Hertzmann. Line drawings from 3d models. *Found. Trends Comput. Graph. Vis.*, 11:1–159, 2019. 4

[4] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 6, 7

[5] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, and Ravi Ramamoorthi. Deep 3d capture: Geometry and reflectance from sparse multi-view images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5960–5969, 2020. 2, 7

[6] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Int. Conf. Comput. Vis.*, pages 12684–12694, 2021. 1, 2

[7] David Bremer and John F. Hughes. Rapid approximate silhouette rendering of implicit surfaces. In *In Proc. Implicit Surfaces*, pages 155–164, 1998. 4

[8] Forrester Cole, Kyle Genova, Avneesh Sud, Daniel Vlasic, and Zhoutong Zhang. Differentiable surface rendering via non-differentiable sampling. In *Int. Conf. Comput. Vis.*, pages 6088–6097, 2021. 2

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 2, 5

[10] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983. 5

[11] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Trans. Graph.*, 33(6):1–12, 2014. 2

[12] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *Proceedings of Machine Learning and Systems*, 2020. 5

[13] John Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12, 06 1995. 4

[14] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 1, 7, 9

[15] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Eur. Conf. Comput. Vis.*, 2016. 8

[16] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. 3

[17] 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

[18] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *SIGGRAPH Asia*, 37(6), 2018. 2, 4

[19] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 5

[20] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *SIGGRAPH Asia*, 38(6), Dec. 2019. 2

[21] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. *Comput. Graph. Forum*, 40(4):101–113, 2021. 2, 3, 4, 5, 6, 7

[22] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4460–4470, 2019. 1, 2

[23] 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 *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 3, 9

[24] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical svbrdf acquisition of 3d objects with unstructured flash photography. *ACM Trans. Graph.*, 37(6):1–12, 2018. 2, 3, 7

[25] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *SIGGRAPH Asia*, 40(6), Dec. 2021. 2

[26] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3504–3515, 2020. 1, 2, 4, 5, 8, 9

[27] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.*, 39(4), July 2020. 2

[28] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 38(6):1–17, 2019. 2, 4

[29] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4531–4540, 2019. 1, 2

[30] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *Int. Conf. Comput. Vis.*, 2021. 1, 2, 3

[31] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 1, 2

[32] 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. *Adv. Neural Inform. Process. Syst.*, 2020. 3, 9

[33] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007. 3

[34] 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. In *Adv. Neural Inform. Process. Syst.*, 2021. 1, 2, 3, 6

[35] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 5, 7

[36] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000. 2

[37] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Trans. Graph.*, 35(6):1–12, 2016. 2

[38] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv preprint arXiv:2106.12052*, 2021. 1, 2, 3

[39] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction with implicit lighting and material. *Adv. Neural Inform. Process. Syst.*, 2020. 1, 2, 3, 4, 5, 7, 8, 9

[40] Cheng Zhang, Zihan Yu, and Shuang Zhao. Path-space differentiable rendering of participating media. *ACM Trans. Graph.*, 40(4):76:1–76:15, 2021. 2, 4, 5, 7

[41] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5453–5462, 2021. 1, 2

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 7

[43] 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. *arXiv preprint arXiv:2106.01970*, 2021. 1, 2