Supplementary Materials for 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

1. 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 [2, 4].

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.

2. Implementation details

We implement our BRDF by following the Mitsuba roughplastic BRDF implementation [1] 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.

3. 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 [3, 5], 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:

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} + t_{\boldsymbol{\Theta}_s} \cdot \boldsymbol{n}. \tag{1}$$

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

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = \boldsymbol{x}. \tag{2}$$

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. 1 into Eq. 2 gives:

$$t_{\boldsymbol{\Theta}_s}\Big|_{\boldsymbol{\Theta}_s = \boldsymbol{\Theta}_s^{(0)}} = 0. \tag{3}$$

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

$$S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}_{\boldsymbol{\Theta}_s}) = 0. \tag{4}$$

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

$$\frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x}_{\boldsymbol{\Theta}_s}) + \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)^T \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\Theta}_s} = 0. \tag{5}$$

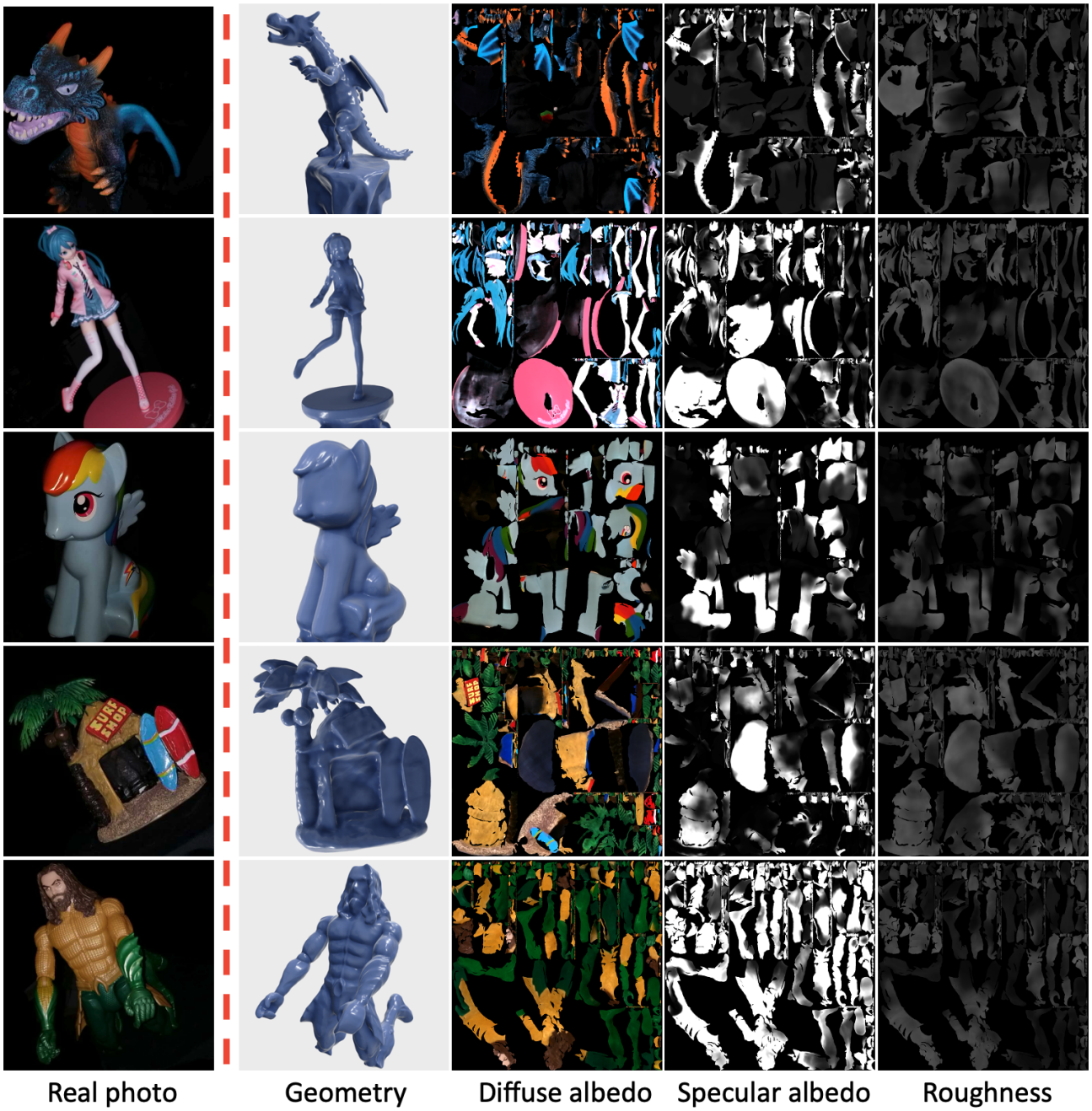

Figure 1. Reconstructed meshes and materials for the real scenes by our IRON system.

Differentiating both sides of Eq. 1 with respect to $\mathbf{\Theta}_s$ gives:

$$\frac{\partial \boldsymbol{x}}{\partial \mathbf{\Theta}_s} = \boldsymbol{n}\frac{\partial t}{\partial \mathbf{\Theta}_s}. \tag{6}$$

Substituting Eq. 6 into Eq. 5 gives us:

$$\frac{\partial S}{\partial \mathbf{\Theta}_s}(\boldsymbol{x}_{\mathbf{\Theta}_s}) + \left(\frac{\partial S}{\partial \boldsymbol{x}}\right)^T \boldsymbol{n}\frac{\partial t}{\partial \mathbf{\Theta}_s} = 0. \tag{7}$$

Evaluating Eq. 7 at $\mathbf{\Theta}_s^{(0)}$, substituting $\boldsymbol{n} = \frac{\partial S}{\partial \boldsymbol{x}}\Big|_{\mathbf{\Theta}_s=\mathbf{\Theta}_s^{(0)}}$ and Eq. 2, we have:

$$\frac{\partial S}{\partial \mathbf{\Theta}_s}(\boldsymbol{x})\Big|_{\mathbf{\Theta}_s=\mathbf{\Theta}_s^{(0)}} + \boldsymbol{n}^T\boldsymbol{n}\frac{\partial t}{\partial \mathbf{\Theta}_s}\Big|_{\mathbf{\Theta}_s=\mathbf{\Theta}_s^{(0)}} = 0. \tag{8}$$

Rearranging a bit, we have:

$$\left.\frac{\partial t}{\partial \boldsymbol{\Theta}_s}\right|_{\boldsymbol{\Theta}_s=\boldsymbol{\Theta}_s^{(0)}} = -\frac{1}{\boldsymbol{n}^T\boldsymbol{n}} \cdot \left.\frac{\partial S}{\partial \boldsymbol{\Theta}_s}(\boldsymbol{x})\right|_{\boldsymbol{\Theta}_s=\boldsymbol{\Theta}_s^{(0)}}. \quad (9)$$

Finally, we conclude our proof by observing that Eqns. 3,9 gives us the first order approximation of $t_{\boldsymbol{\Theta}_s}$:

$$t_{\boldsymbol{\Theta}_s} = -\frac{1}{\boldsymbol{n}^T\boldsymbol{n}} \cdot S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}). \quad (10)$$

Substituting Eq. 10 into Eq. 1 leads to our proposed edge point re-parametrization:

$$\boldsymbol{x}_{\boldsymbol{\Theta}_s} = \boldsymbol{x} - \frac{\boldsymbol{n}}{\boldsymbol{n}^T\boldsymbol{n}} \cdot S_{\boldsymbol{\Theta}_s}(\boldsymbol{x}). \quad (11)$$

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.

4. Reconstructed meshes and materials

In Fig. 1, we show the reconstructed meshes and materials for the 5 real-world scenes used in this work.

5. 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. 2 and the caption for an example comparison.

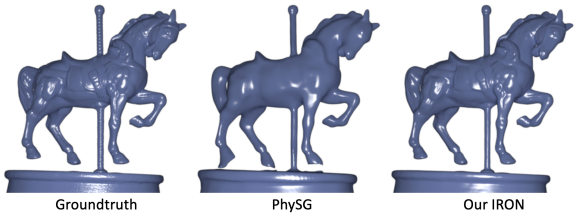

Figure 2. For the horse object, our method recovers much more accurate geometry details than PhySG; chamfer L1 distances are: Our IRON (5.35e-4) vs. PhySG (18.67e-4).

References

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

[2] 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

[3] 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

[4] 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. 1

[5] 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