Supplementary of Physics-based Indirect Illumination for Inverse Rendering

Youming Deng
Cornell University

Xueting Li
NVIDIA Research

Sifei Liu
NVIDIA Research

Ming-Hsuan Yang
UC Merced

In this supplementary material, we add more qualitative comparisons on both synthetic and real-world datasets in Sec. 1. After that, we introduce the Spherical Gaussians (SGs) function, the multiplications of SGs in Sec. 2, and how to use SGs to represent terms in the rendering equation in Sec. 3. Then we prove how to solve the non-differentiability at boundary lights on one dimension and use the Reynolds transport theorem [11] to extend Leibniz integral rule to a higher dimension in Sec. 4. The details of network architecture and modification of the decoder are introduced in Sec. 5. Next, we show comprehensive ablation studies in Secs. 6 and 7. Finally, we discuss differences between the proposed method and SOTA baselines as well as the limitations of our method in Sec. 8 and Sec. 9, respectively. We encourage the reviewers to check our accompanying videos for an overview of the proposed method and more results.

1. Extra Qualitative Comparisons

We show two more qualitative real-world and synthetic datasets comparisons in Fig. 2 and Fig. 1, respectively. Similar to the comparison of synthetic dataset in the main paper, our method is capable of predicting more accurate material and optimized illumination (See Fig. 1). As for the real-world dataset, the baselines [17, 19] cannot accurately disentangle albedo and roughness, while our method has reasonable decomposition (See Fig. 2).

2. Spherical Gaussians Function

2.1. Representation of SGs

A general form of SGs can be represented as:

$$G(\boldsymbol{x};\boldsymbol{\xi},\lambda,\boldsymbol{\mu}) = \boldsymbol{\mu} e^{\lambda(\boldsymbol{x}\cdot\boldsymbol{\xi}-1)}, \tag{1}$$

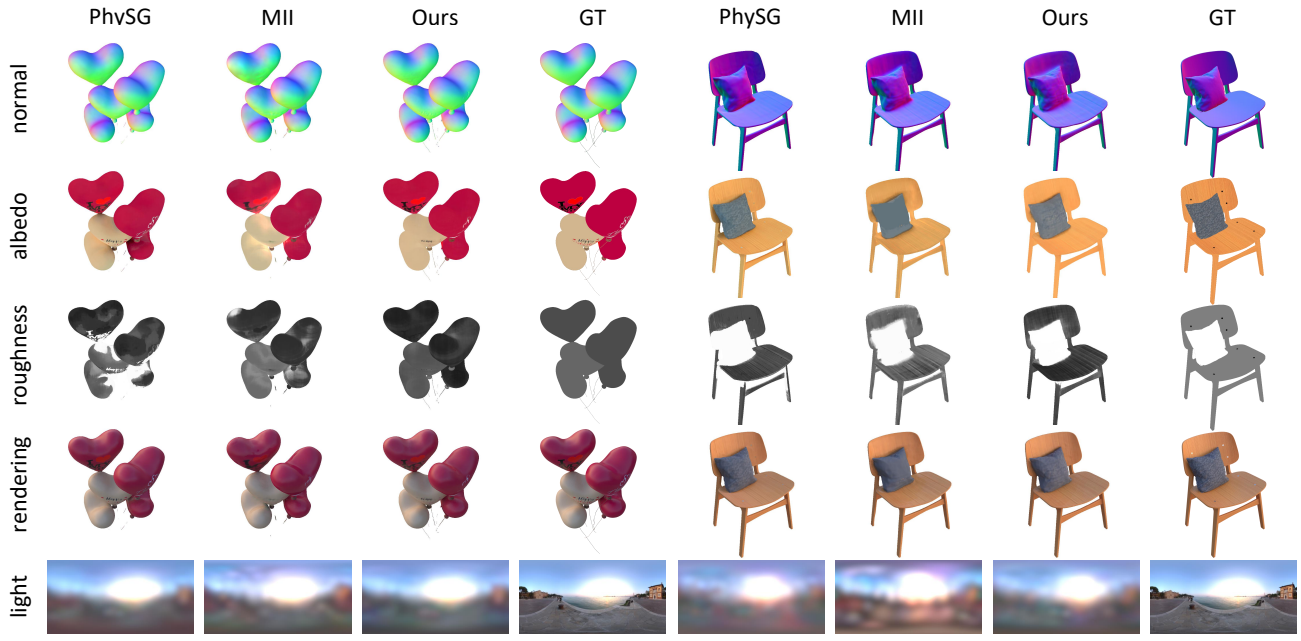


Figure 1. **Qualitative Comparisons on Synthetic Dataset**. We compare our method with baselines on hotdog and chair scene.

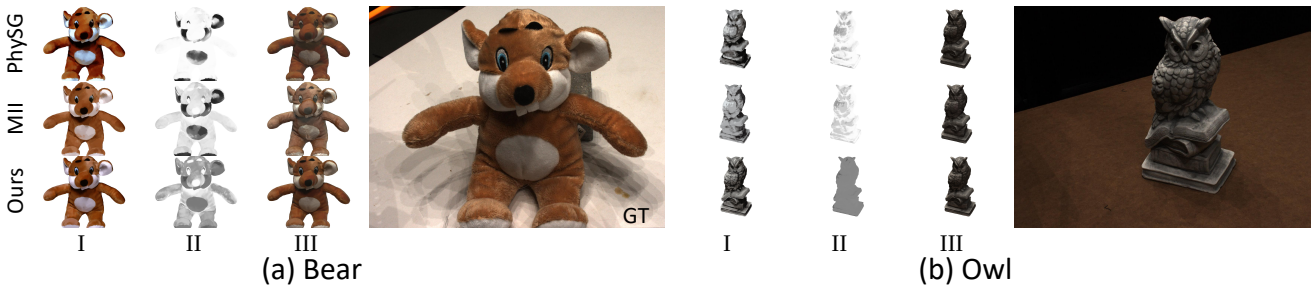


Figure 2. **Qualitative Comparisons on DTU** [5]. We compare our method with baselines on toy bear and owl statue.

where $\boldsymbol{x} \in \mathbb{S}^2$ is the input of SGs function, $\boldsymbol{\xi} \in \mathbb{S}^2$ is the lobe axis, $\lambda \in \mathbb{R}_+$ is the lobe sharpness, and $\boldsymbol{\mu} \in \mathbb{R}^3$ is the lobe amplitude.

2.2. Multiplication of SGs

Following [14], we represent the multiplication of two different SGs $G(\boldsymbol{x}; \boldsymbol{\xi}_1, \lambda_1, \boldsymbol{\mu}_1)$ and $G(\boldsymbol{x}; \boldsymbol{\xi}_2, \lambda_1, \boldsymbol{\mu}_1)$ using a third one $G(\boldsymbol{x}; \boldsymbol{\xi}', \lambda', \boldsymbol{\mu}')$:

$$\begin{aligned} G_1 \cdot G_2 &= \boldsymbol{\mu}_1 e^{\lambda_1(\boldsymbol{x}\cdot\boldsymbol{\xi}_1 - 1)} \cdot \boldsymbol{\mu}_2 e^{\lambda_2(\boldsymbol{x}\cdot\boldsymbol{\xi}_2 - 1)} \\ &= \boldsymbol{\mu}' e^{\lambda'(\boldsymbol{x}\cdot\boldsymbol{\xi}' - 1)}, \end{aligned} \tag{2}$$

where

$$\boldsymbol{\mu}' = \boldsymbol{\mu}_1 \boldsymbol{\mu}_2 e^{\lambda_3 - \lambda_1 - \lambda_2}, \tag{3}$$

$$\lambda' = \lambda_3 = ||\lambda_1 \boldsymbol{\xi}_1 + \lambda_2 \boldsymbol{\xi}_2||, \tag{4}$$

$$||\lambda_1 \boldsymbol{\xi}_1 + \lambda_2 \boldsymbol{\xi}_2|| = \sqrt{\lambda_1^2 + \lambda_2^2 + 2\lambda_1\lambda_2(\boldsymbol{\xi}_1 \cdot \boldsymbol{\xi}_2)}, \tag{5}$$

$$\boldsymbol{\xi}' = \frac{\lambda_1 \boldsymbol{\xi}_1 + \lambda_2 \boldsymbol{\xi}_2}{\lambda_3}. \tag{6}$$

3. Rendering Equation Representation

The rendering equation [6] used in the submission is:

$$L_o(\boldsymbol{\omega}_o; \boldsymbol{x}) = \int_\Omega L_i(\boldsymbol{\omega}_i) f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \boldsymbol{x})(\boldsymbol{\omega}_i \cdot \boldsymbol{n}) \mathrm{d}\boldsymbol{\omega}_i. \tag{7}$$

where $L_i(\boldsymbol{\omega}_i)$ is the incident light, $f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \boldsymbol{x})$ is simplified Disney BRDF model [1] at the surface point $\boldsymbol{x}$ and $\boldsymbol{n}$ is the normal at this point.

We discuss how to represent each part in the rendering equation using SGs. It is worth noticing that we only consider *a single incidence direction* $\boldsymbol{\omega}_i$ in the following part for simplicity.

3.1. Incident (Environment) Light

We simulate continuous environment illumination as composed of different discrete light sources:

$$L_i(\boldsymbol{\omega}_i) = \boldsymbol{\mu} e^{\lambda(\boldsymbol{\omega}_i \cdot \boldsymbol{\xi} - 1)}, \tag{8}$$

where $\boldsymbol{\omega_i} \in \mathbb{S}^2$ is the input of incident light SGs function, $\boldsymbol{\xi} \in \mathbb{S}^2$ is the lobe axis, $\lambda \in \mathbb{R}_+$ is the lobe sharpness, and $\boldsymbol{\mu} \in \mathbb{R}^3$ is the lobe amplitude.

3.2. Representing $\boldsymbol{\omega}_i \cdot \boldsymbol{n}$ as SGs

The multipliciation of the view direction $\boldsymbol{\omega}_i$ and surface normal $\boldsymbol{n}$ can also be transformed into SGs [10] as:

$$\begin{aligned} \boldsymbol{\omega}_i \cdot \boldsymbol{n} &= G(\boldsymbol{\omega}_i; 0.0315, \boldsymbol{n}, 32.7080) - 31.7003 \\ &= 32.7080 \cdot e^{\boldsymbol{n}(\boldsymbol{\omega}_i \cdot 0.0315 - 1)} - 31.7003. \end{aligned} \tag{9}$$

3.3. Specualr Term in BRDF

The simplified Disney BRDF model [1] split reflectance into diffuse and specular:

$$f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \boldsymbol{x}) = \frac{a}{\pi} + f_s(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \boldsymbol{x}) \tag{10}$$

We have explained the first term in the submission, and the second term f_s is modeled with the Microfacet Model [3, 13]:

$$f_s(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \boldsymbol{x}) = \frac{F(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)\mathbf{G}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)D(\boldsymbol{h})}{4 \cdot (\boldsymbol{n} \cdot \boldsymbol{\omega}_o)(\boldsymbol{n} \cdot \boldsymbol{\omega}_i)}, \tag{11}$$

There are three important terms $(F, \mathbf{G}, D)$ to simulate specular:

- Fresnel term F that evaluates how much light is reflected by the surface under a given angle of incidence.
- Geometric attenuation term $\mathbf{G}$ (*i.e.*, masking and shadowing term) that accounts for mutual shadowing and masking of microfacets.
- A microfacet distribution function D that tells what fraction of microfacets are oriented in direction $\boldsymbol{h} = \frac{\boldsymbol{\omega}_o + \boldsymbol{\omega}_i}{||\boldsymbol{\omega}_o + \boldsymbol{\omega}_i||_2}$ so that light incoming from direction $\boldsymbol{\omega}_i$ will be reflected in direction $\boldsymbol{\omega}_o$.

We represent each term as in [7]:

$$F(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i) = \boldsymbol{S} + (1 - \boldsymbol{S}) \cdot 2^{-(5.55473\boldsymbol{\omega}_o \cdot \boldsymbol{h} + 6.8316)(\boldsymbol{\omega}_o \cdot \boldsymbol{h})}, \tag{12}$$

$$\mathbf{G}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i) = \frac{\boldsymbol{\omega}_o \cdot \boldsymbol{n}}{\boldsymbol{\omega}_o \cdot \boldsymbol{n}(1-k) + k} \cdot \frac{\boldsymbol{\omega}_i \cdot \boldsymbol{n}}{\boldsymbol{\omega}_i \cdot \boldsymbol{n}(1-k) + k}, \tag{13}$$

$$D(\boldsymbol{h}) = G(\boldsymbol{h}; \boldsymbol{n}, \frac{2}{R^4}, \frac{1}{\pi R^4}) = \frac{1}{\pi R^4} e^{\frac{2}{R^4}(\boldsymbol{h} \cdot \boldsymbol{n} - 1)}, \tag{14}$$

where $\boldsymbol{S}$ is the specular reflectance (Fresnel coefficient), R is the roughness, and $k = \frac{(R+1)^2}{8}$.

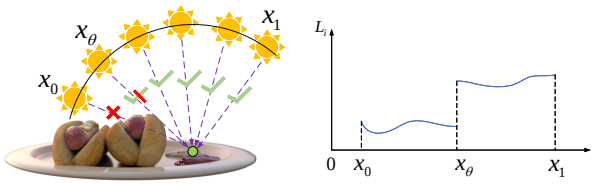


Figure 3. **Demonstration of Non-differentiability at Boundary Lights**. We borrow the figure in the main body here for the explanation.

4. Non-differentiability at Boundary Lights

To better understand the high-level idea, we simplify the derivative of an evolving surface [2] with non-motion assumption [16] and present the partial gradient computation deduction for the boundary lights in one dimension. The environment lights are all within a 2D plane and uniformly distributed between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, and $\boldsymbol{x}_\theta$ is the boundary light that $L_i(\boldsymbol{x}_i)$ change abruptly (*i.e.*, from "obstructed" to "visible", see the supplementary video for animation) and the location is related to scene parameters θ. The rendering equation can be simplified to:

$$L_o(\boldsymbol{\omega}_o; \boldsymbol{p}) = \int_{\boldsymbol{x}_0}^{\boldsymbol{x}_1} L_i(\boldsymbol{x}_i) f_r(\boldsymbol{\omega}_o, \boldsymbol{x}_i; \boldsymbol{p})(\boldsymbol{x}_i \cdot \boldsymbol{n}) \mathrm{d}\boldsymbol{x}_i, \quad (15)$$

where $\boldsymbol{p}$ represents location of green point Fig. 3 and its surface normal $\boldsymbol{n}$. We replace $L_i(\boldsymbol{x}_i) f_r(\boldsymbol{\omega}_o, \boldsymbol{x}_i; \boldsymbol{p})(\boldsymbol{x}_i \cdot \boldsymbol{n})$ using $F(\boldsymbol{x}_i)$ for the clean deduction.

When calculating the partial derivative of $L_o(\boldsymbol{\omega}_o; \boldsymbol{p})$ w.r.t. to network parameters $\boldsymbol{\theta}$, we can get:

$$\begin{aligned}
&\frac{\partial L_o(\boldsymbol{\omega}_o; \boldsymbol{p})}{\partial \theta} = \frac{\partial}{\partial \theta} \int_{\boldsymbol{x}_0}^{\boldsymbol{x}_1} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x}_i \\
&= \frac{\partial}{\partial \theta} \int_{\boldsymbol{x}_0}^{\boldsymbol{x}_\theta - \boldsymbol{\epsilon}} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x} + \frac{\partial}{\partial \theta} \int_{\boldsymbol{x}_\theta + \boldsymbol{\epsilon}}^{\boldsymbol{x}_1} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x} \\
&= F(\boldsymbol{x}_\theta - \boldsymbol{\epsilon}) \cdot \frac{\partial \boldsymbol{x}_\theta}{\partial \theta} + \int_{\boldsymbol{x}_0}^{\boldsymbol{x}_\theta - \boldsymbol{\epsilon}} \frac{\partial}{\partial \theta} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x} \\
&- F(\boldsymbol{x}_\theta + \boldsymbol{\epsilon}) \cdot \frac{\partial \boldsymbol{x}_\theta}{\partial \theta} + \int_{\boldsymbol{x}_\theta + \boldsymbol{\epsilon}}^{\boldsymbol{x}_1} \frac{\partial}{\partial \theta} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x} \\
&= \Delta F(\boldsymbol{x}_\theta) \cdot \frac{\partial \boldsymbol{x}_\theta}{\partial \theta} + \int_{\boldsymbol{x}_0 \boldsymbol{x}_1 - \boldsymbol{x}_\theta} \frac{\partial}{\partial \theta} F(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x}
\end{aligned} \quad (16)$$

where $\boldsymbol{x}_0 \boldsymbol{x}_1 - \boldsymbol{x}_\theta$ means removing $\boldsymbol{x}_\theta$ at the edge.

We can extend this to two-dimensional rendering integral, using Reynolds transport theorem [11]. A comprehensive high dimensional proof can be found in [4, 16].

5. Detailed Network Architecture

Fig. 4 visualizes the detailed architecture. For each network, the yellow blocks are the inputs, and the orange blocks are the outputs.

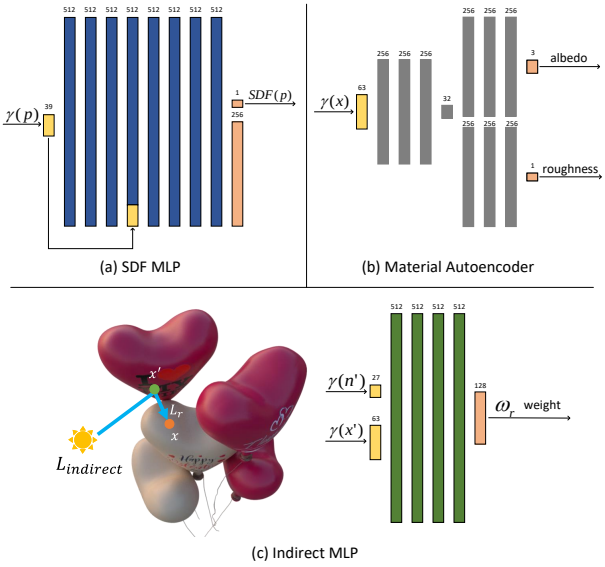


Figure 4. **Detailed Architecture.** (a) The SDF MLP in Sec. 4.1 in the main paper. (b) The material autoencoder in Sec. 4.4 in the main paper. (c) The indirect MLP in Sec. 4.3.1 in the main paper. $\gamma(\cdot)$ indicates the positional encoding procedure.

5.1. Network Design

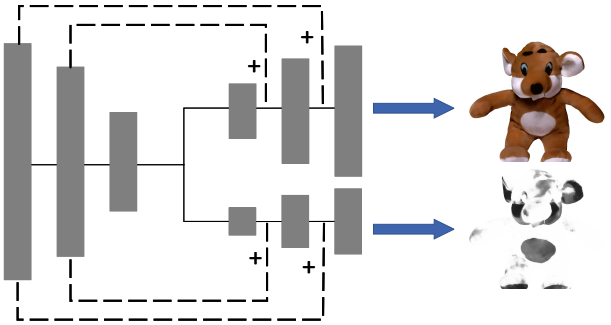
Fig. 4a shows the SDF MLP (F_{SDF} in Sec. 4.1 in the main paper). It takes the positional embedding of a query 3D point $\boldsymbol{p}$, and outputs its SDF value and a 256 dimension feature embedding (used for geometry initialization stage as [19]). The SDF MLP uses the softplus activation after each hidden layer. Fig. 4b illustrates the material autoencoder (Sec. 4.4 in the main paper). It takes the positional embedding of a surface point $\boldsymbol{x}$ as input and predicts the albedo and roughness at this point. Fig. 4c demonstrates the indirect MLP (F_{ind} in Sec. 4.3.2 in the main paper). To estimate the target indirect light L_r, the F_{ind} takes the positional embedding of a side traced point $\boldsymbol{x}'$ (the intersection of L_r and an object when reaching light sources) and its normal $\boldsymbol{n}'$ as inputs and predict the environment lights weights $\boldsymbol{w_r}$. We then utilize Eq. (5) in the main paper to aggregate the environment lights w.r.t. $\boldsymbol{w_r}$ and predict the target indirect light L_r. Both the material autoencoder and indirect MLP use ReLU as an activation function after each hidden layer.

5.2. Decoder Architecture on Real-world Datasets

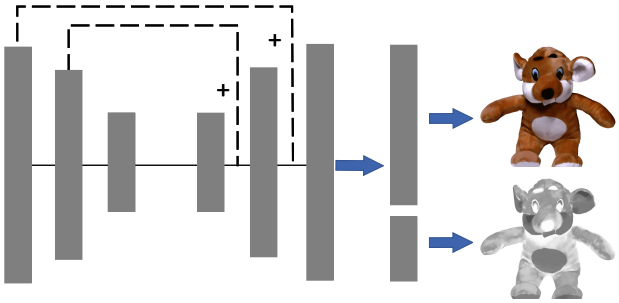
The material autoencoder described in Sec. 5.1 works well on the synthetic dataset. However, we observe that when applying it to a real-world dataset, it is prone to trivial solutions (e.g., the saturated roughness in Fig. 5). To resolve this issue, we share more layers between the albedo and

Figure 5. **Qualitative Comparison of Two Decoders on DTU** [5]. Using separate decoders makes the model predict saturated roughness. This phenomenon can be resolved by sharing more layers between the roughness and albedo decoder.

roughness decoders, as shown in Fig. 6 (b). Intuitively, this design adds a mutual constraint on roughness and albedo estimation as a regularization to prevent the roughness from saturating. As shown in Fig. 2, our method can decompose each scene into reasonable albedo, roughness, and geometry normal using this design.



(a) Two Separate Decoders



(b) Shared Decoder

Figure 6. **Different Material Decoders**. We visualize the structure of two different decoders. For real-world datasets, sharing more layers (b) for decoding can add regularization on roughness prediction to avoid trivial solutions occurring in (a).

6. Ablations on Tracing Refinement

We show the time cost in Tab. 1. Refined sphere tracing allows a larger threshold (10^{-3} compared to 10^{-5} in classic sphere tracing) without compromising tracing accuracy, which accelerates the geometry initialization training significantly from 22 hours to 17 hours. However, the refinement requires gradient computation (Eq. (4) in the main paper), which cannot be done simultaneously for many pixels during inference due to memory limitation. So classic sphere tracing is used during inference and causes a longer inference time. We will further explore how this computationally expensive tracing under a large number of pixels can be applied.

To further demonstrate the effectiveness of the refinement process, we perform ablation studies by removing this refinement process. The results are shown in Fig. 7. Without this refinement process under the threshold of 10^{-3}, the model cannot capture the geometry details in the hot dog scene, the ribbons below the balloons, and the surface of the balloons present noticeable artifacts.

7. Ablations on Boundary Lights

Despite the quantitative evaluation in Tab. 1d of the main body, We visualize the qualitative results of including boundary lights or not. Though the inverse rendering results (*i.e.*, albedo and roughness estimation) are similar, our full model achieves much better light estimation results, as demonstrated in Fig. 9. Boundary lights are crucial to estimate realistic environment lights. Without modeling boundary lights, the predicted environment lights are noisy and inconsistent.

8. Detailed Comparison with SOTA Methods

In this section, we compare our method and PhySG [17], MII [19], NeRV [12], respectively. We discuss the details below.

PhySG [17] resolves inverse rendering by modeling illumination as mixtures of Spherical Gaussians combined with an implicit function for geometry estimation. Our method has two main advantages compared to PhySG [17]: a) our approach explicitly models indirect illumination that is ignored in PhySG. b) PhySG [17] assumes that the object surface has a consistent roughness, which is often not applicable in relatively cluttered scenes that include more than one object. At the same time, our method predicts a spatially-varying roughness that is more capable for complex scenes.

MII [19] (as well as several other works [9, 12, 15, 18]) models indirect illumination by learning an MLP that takes a surface point (e.g., $\boldsymbol{x}$ in Fig. 4(c)) as input and directly outputs the indirect illumination at the point. The indirect illumination prediction accuracy is thus limited by the capacity of the MLP. On the other hand, our method explicitly traces and analyzes each indirect light at a surface point, leading to more precise indirect light illumination.

Finally, compared to NeRV [12], our model has the following advantages: a) our approach does not require known

Method	Stage	GPU / Peak Memory	Training Time / Tracing Threshold	Inference Time / Tracing Threshold
MII [42]	Geometry Initialization	GeForce GTX 1080 Ti / ≈ 2GB	22 hours / 10^{-5}	around 50s per img / 10^{-5}
	Illumination	NVIDIA RTX A5000 / ≈ 20GB	1.5 hours / 10^{-5}	
	Illumination & Material	NVIDIA RTX A5000 / ≈ 20GB	1 hours / 10^{-5}	
Ours	Geometry Initialization	GeForce GTX 1080 Ti / ≈ 2GB	17 hours / 10^{-3}	around 120s per img / 10^{-5}
	Geometry & Illumination & Material	NVIDIA RTX A5000 / ≈ 20GB	6 hours / 10^{-3}	

Table 1. **Analysis on Speed and Threshold.**

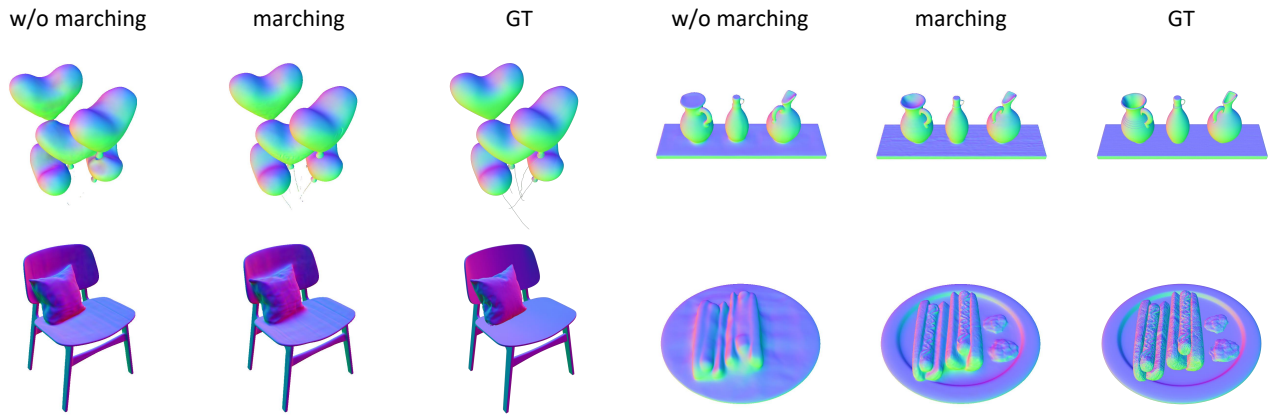


Figure 7. **Ablations on Tracing Refinement**.

environment light. Thus can be easily applied to real-world scenes, as shown in Fig. 5. b) NeRV employs an integral that uniformly aggregates environment lights for indirect illumination computation, which is equivalent to the uniform weight strategy discussed in Sec.5.3 in the paper. As shown in Fig.4 in the paper, this uniform weight strategy leads to compromised scene reconstruction (i.e., inferior quantitative results in Tab. 1 in the paper and shading area in Fig. 4b). c) NeRV locates less accurate reflection point locations through depth prediction by an MLP rather than physical tracing.

Figure 8. **Failed Case**.

9. Discussions and Limitations

Self-supervised inverse rendering is a highly ill-posed problem; in rare cases, it suffers from accurately disentangling geometry and texture. As shown in Fig. 6 of the main paper, the strips on the board under the jars can be either interpreted as grooves (as our model did) or the texture patterns of the board (as in the ground truth images). Resolving such ambiguity requires either ground truth supervision or prior knowledge. Furthermore, thanks to the physics-based design, our method naturally supports relighting. While the results in Fig.8 (b) demonstrate a high level of realism, they may not align perfectly with the ground truth. We attribute it to the ill-posedness of self-supervised inverse rendering discussed above. Specifically, our training data only comprises images captured under certain environment lights, causing F_{ind} (See Sec.4.3.2 in the paper) to learn weights for indirect light combination under this specific environment illumination. This issue could be mitigated if images captured under different illuminations are available.

Following prior methods [12, 15, 17, 19], in this work, we focus on non-specular objects. As a result, our method can not accurately predict the inverse rendering of a strong reflection object (though we can edit the surface of an object to have strong reflection, as shown in Fig.8 (c) in the main paper). A potential solution is to pre-train the model on an annotated dataset to learn a prior of specular objects [8].

In the fine-tuning stage, our approach exhibits a slower optimization compared to the MII [19]. This reduced speed stems from the necessity of a secondary reverse ray marching process, a critical step to precisely identify reflection points. This choice is pivotal for accurately modeling environmental illumination. Consequently, while there is an increase in computation time, this trade-off is justified by the enhanced accuracy in material and illumination recovery.

Last, the SDF MLP used in our method can not capture highly delicate geometry such as hairs or furs. As is shown in Fig. 8, the beard of the Smurf degrades into a flat plane, leading to failed disentanglement of albedo and roughness.

We leave these limitations to future work.

10. Video Demos

We highly encourage the reviewers to watch our video in the zip file. In the video, we demonstrate details of the proposed method and show more comprehensive inverse rendering as well as editing results on four synthetic scenes.

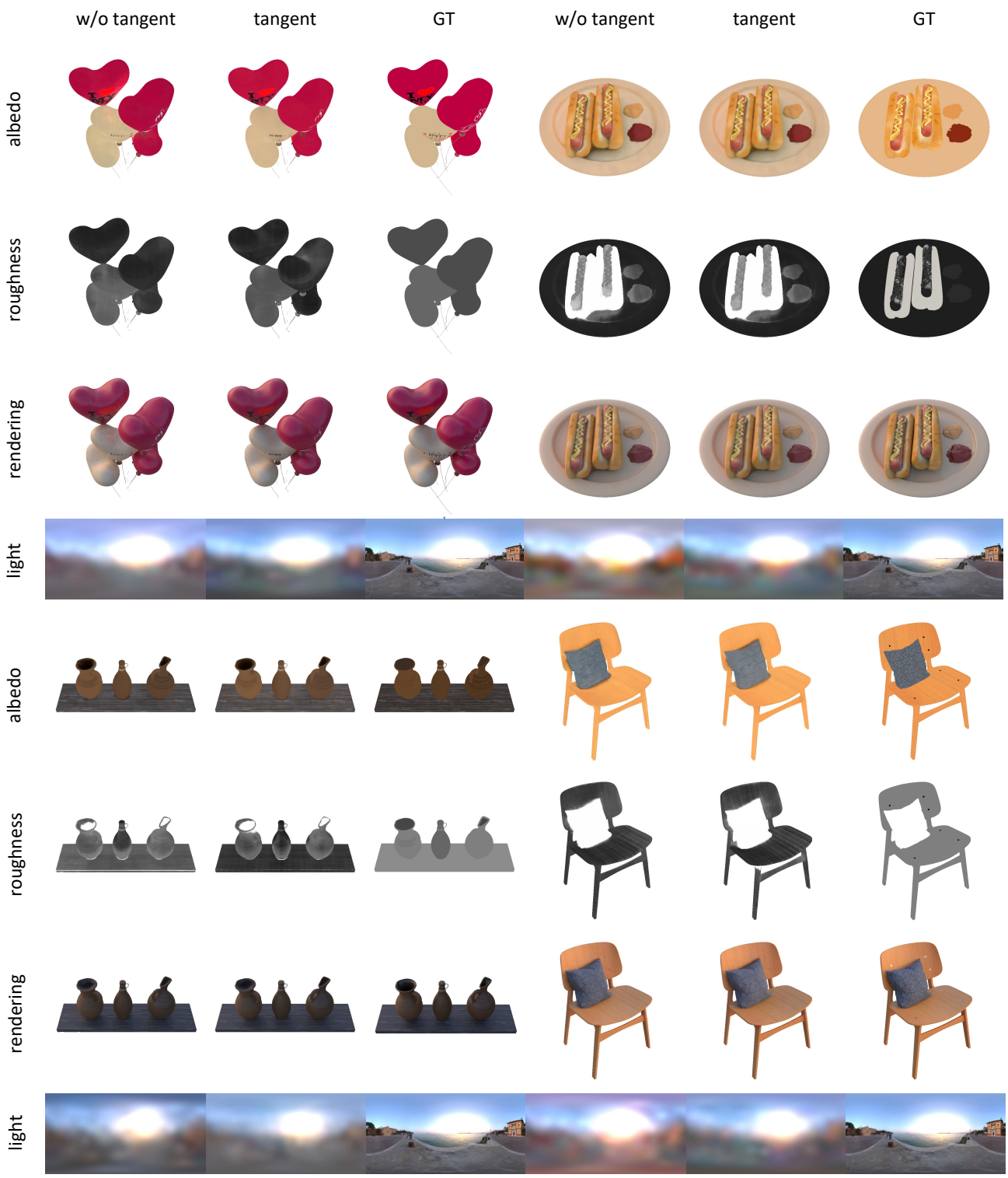


Figure 9. **Ablation Study on Boundary Lights**. Our full model can achieve better light estimation results by considering boundary lights.

References

[1] Brent Burley and Walt Disney Animation Studios. Physically-based shading at Disney. In *SIGGRAPH*, 2012. 2

[2] Paolo Cermelli, Eliot Fried, and Morton E Gurtin. Transport

relations for surface integrals arising in the formulation of balance laws for evolving fluid interfaces. *Journal of Fluid Mechanics*, 2005. 3

[3] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM ToG*, 1982. 2

[4] Harley Flanders. Differentiation under the integral sign. *The American Mathematical Monthly*, 1973. 3

[5] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 2, 4

[6] James T. Kajiya. The rendering equation. In *SIGGRAPH*, 1986. 2

[7] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM ToG*, 2018. 2

[8] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. *ACM ToG*, 2023. 6

[9] Linjie Lyu, Marc Habermann, Lingjie Liu, Mallikarjun B. R., Ayush Tewari, and Christian Theobalt. Efficient and differentiable shadow computation for inverse problems. In *ICCV*, 2021. 5

[10] Julian Meder and Beat D. Brüderlin. Hemispherical gaussians for accurate light integration. In *ICCVG*, 2018. 2

[11] Osborne Reynolds. *Papers on mechanical and physical subjects: the sub-mechanics of the universe*. University Press, 1903. 1, 3

[12] 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 *CVPR*, 2021. 5, 6

[13] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *EGSR*, 2007. 2

[14] Kun Xu, Yan-Pei Cao, Li-Qian Ma, Zhao Dong, Rui Wang, and Shi-Min Hu. A practical algorithm for rendering all-frequency interreflections. *ACM ToG*, 2014. 2

[15] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K. Wong. Ps-nerf: Neural inverse rendering for multi-view photometric stereo. In *ECCV*, 2022. 5, 6

[16] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A differential theory of radiative transfer. *ACM ToG*, 2019. 3

[17] 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 *CVPR*, 2021. 1, 5, 6

[18] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul E. Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM ToG*, 2021. 5

[19] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, 2022. 1, 3, 5, 6