Supplementary Material: Neural Fields meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes

Zian Wang[1,2,3] Tianchang Shen[1,2,3] Jun Gao[1,2,3] Shengyu Huang[1,4] Jacob Munkberg[1]
Jon Hasselgren[1] Zan Gojcic[1] Wenzheng Chen[1,2,3] Sanja Fidler[1,2,3]
[1]NVIDIA [2]University of Toronto [3]Vector Institute [4]ETH Zürich

In the supplementary material, we first provide details on our model design choices (Sec. A). Then we describe the training details (Sec. B). Finally, we provide experiment details and additional results (Sec. C). Please refer to the accompanied video for qualitative results on relighting and virtual object insertion.

A. Model Details

Geometry definition. Our method relies on an explicit surface definition for mesh extraction and efficient ray tracing. To this end, we follow NeuS [25] and model the geometry with a Signed Distance (SD) Field whose zero-level set defines the scene surface. For volume rendering, the SDF values $f_{\text{SDF}}(\mathbf{r}(t))$ are converted to opacity densities $\rho(\mathbf{r}(t))$ as:

$$\rho(\mathbf{r}(t)) = \max\left(\frac{-\frac{\mathrm{d}\Phi_\kappa}{\mathrm{d}t}(f_{\text{SDF}}(\mathbf{r}(t)))}{\Phi_\kappa(f_{\text{SDF}}(\mathbf{r}(t)))}, 0\right) \tag{1}$$

where $\Phi_\kappa(x) = \text{Sigmoid}(\kappa x) = \frac{1}{1+e^{-\kappa x}}$. Intuitively, the conversion is approximated by placing a unimodal function around the zero-level set of the SD field, *i.e.* the derivative of the sigmoid function $\Phi_\kappa^{'}(x)$. Here, κ is a learnable parameter that controls the sharpness of the function and empirically $1/\kappa$ converges to zero as the training proceeds [25]. To extract the mesh, we run marching cubes by querying the SD field on a predefined grid.

Material definition. We define the material properties of the scene using the physically-based (PBR) material model from Disney [5], which is a standard BRDF model adopted by modern graphics engines such as Unreal Engine [9].

The PBR material model represents the material properties using a 3-channel base color $\mathbf{k}_d \in \mathbb{R}^3$, and 2-channel specular properties $\mathbf{k}_s \in \mathbb{R}^2$. Here, $\mathbf{k}_s$ includes the roughness and metallic parameters. The metallic parameter is a real value $\in [0, 1]$ indicating whether the surface behaves as a metal or nonmetal surface (e.g., plastic). Similarly, the roughness parameter is also a real value $\in [0, 1]$ and defines how rough or smooth the surface is, thereby controlling how sharp or blurry reflections appear on that surface.

In Fig. 1, 2 and 5 of the main paper, we visualize linear base color as an RGB image, and follow the graphics convention to visualize the specular properties $\mathbf{k}_s$ as a packed RGB image, where metallic is visualized with R-channel and roughness with G-channel.

Normal extraction. Recent works have proposed different ways to extract normal vectors from a neural field. For example, IRON [29] directly used the gradient of the underlying SD field, NeROIC [11] used volume convolution, and Ref-NeRF [24] introduced an MLP network that predicts a normal vector at each point to regularize the noisy gradients of their volume density field.

In our method, we use an MLP $f_{\text{norm.}}$ to predict the normal direction for any 3D location, and estimate the normal vectors through volume rendering of the normal field. We further regularize the predicted normal directions to be consistent with normals computed from the gradient of SD Field. This design choice allows us to softly enforce the consistency with the underlying SDF geometry, while still maintaining the flexibility to account for high-frequency shading details with an MLP predicted normal (similar to a normal bump map in mesh-based representation [7]).

Exposure and HDR to LDR conversion. Real-world cameras often perform automatic white balancing and exposure correction [20] that result in inconsistent supervision signals across the input images. To alleviate this issue, we additionally optimize a per-image exposure. Specifically, we optimize a set of variables $\{\beta_i\}_{i=1}^N$, where $\beta_i \in \mathbb{R}^3$ corresponds to the $i-$th image exposure compensation. During optimization, we normalize β_i over all the images to resolve scale ambiguity: $\beta_i = \frac{\tilde{\beta}_i}{\frac{1}{N}\sum_{i=1}^N \tilde{\beta}_i}$. After that, we multiply the exposure compensation ratio β_i with the predicted RGB values in the HDR linear RGB space. During inference, we set all β_i to a vector of ones. Note that our lighting intensity and the rendering output of each pixels are HDR values in linear RGB space, while the ground truth values of each pixel in the captured image are in LDR sRGB space. To convert the predicted HDR RGB values to LDR sRGB, we

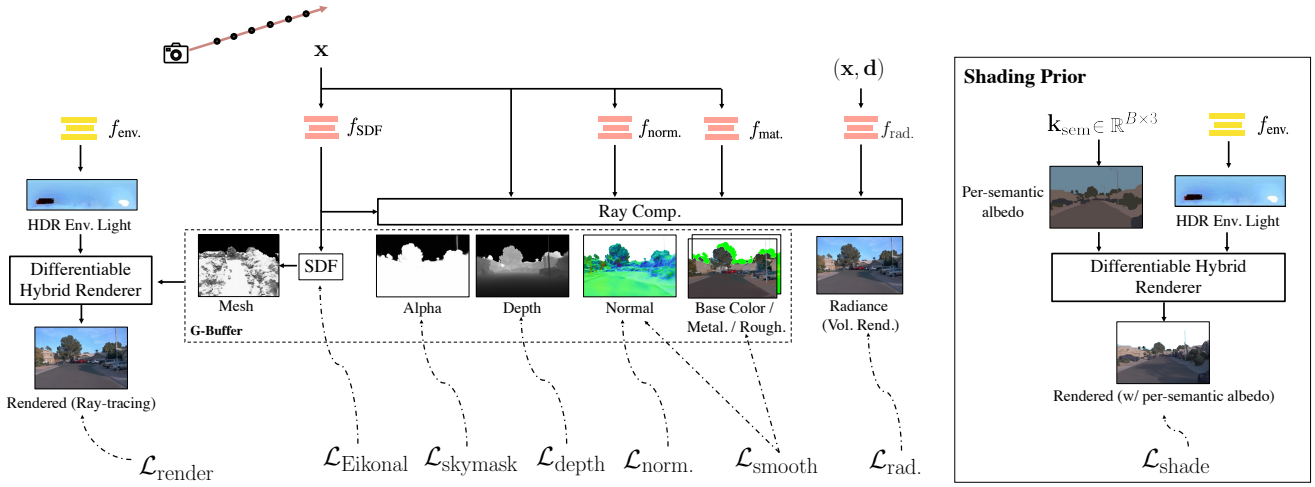

Figure A. **Training details for FEGR**. Similar to prior optimization-based inverse rendering methods [1, 4, 7], FEGR adopts the major supervision of image reconstruction loss $\mathcal{L}_{\text{render}}$ and a set of regularization terms for each intrinsic property.

use a standard gamma correction (gamma value equals to 2.2) and intensity clipping [13, 27].

Implementation details. The networks f_{SDF}, $f_{\text{norm.}}$ and $f_{\text{mat.}}$ are 2-layer MLPs with a multi-resolution hash positional encoding [16], representing SDF, surface normal and material properties respectively. The dimension of the hidden layer is 64. The network $f_{\text{env.}}$ is 4-layer MLP with frequency positional encoding [15] and exponential activation, representing HDR environment lighting. The hidden layer dimension is 256. For each primary ray, we sample 512 uniformly-spaced points and 64 adaptively sampled points following the scheme of NeuS [25]. We sample 512 secondary rays via importance sampling over the BRDF and the HDR environment map. We extract the mesh using marching cubes [14] with a $512 \times 512 \times 64$ grid, implemented in PyTorch [19] with CUDA support. Adapted from Nvdiffrecmc [7], the differentiable shading module is implemented in CUDA with OptiX [18]. In the backward pass, we stop the gradient back-propagation to the extracted triangle meshes due to the GPU memory constraints. The inference time for marching cubes mesh extraction is 130ms. After each mesh update, the time to rebuild the bounding volume hierarchy (BVH) [18] is 75ms. Note that the mesh extraction and BVH are only computed once per scene during inference. The shading pass of one 640x960 image takes 210ms.

B. Training Details

In the following, we provide additional details for each loss function, as well as an intuitive explanation of their contribution to the combined optimization. An overview of our training pipeline is provided in Fig. A. Except for shading prior loss $\mathcal{L}_{\text{shade}}$, similar loss terms were used before in the literature. The ablation study provided in Sec. C and Fig B therefore focuses on the $\mathcal{L}_{\text{shade}}$.

Shading prior $\mathcal{L}_{\text{shade}}$. As is described in the main paper in Sec. 3.3, the motivation for introducing the semantics-aware shading regularization term $\mathcal{L}_{\text{shade}}$ is to regularize the *lighting*. Indeed, $\mathcal{L}_{\text{shade}}$ encourages that the shadows present in the input images are explained by the combination of lighting and geometry, instead of degenerating into an easy solution of baking them into albedo.

To this end, we introduce an auxiliary piecewise-constant albedo representation and encourage its re-rendering to be consistent with the groundtruth image. Intuitively, due to the limited capacity of the piecewise-constant albedo representation, the supervision signal emerging from the lighting effects will be mainly propagated to the HDR environment light. Specifically, we initialize each semantic class with a 3-channel albedo value, which we optimize during training. Thereby semantic segmentation labels are computed with an off-the-shelf semantic segmentation network [23]. To compute the $\mathcal{L}_{\text{shade}}$, we use the estimated lighting to render this per-semantic class albedo and encourage the rendered result to be consistent with the groundtruth images.

We depict an example of the per-semantic class albedo in Fig. A (right). In practice, we apply this loss on the semantic classes *road, sidewalk, building, wall* which typically have a single dominant albedo and provide informative visual cues such as boundary of cast shadows. We ablate the effect of this loss in Sec. C and Fig B.

Regularization terms $\mathcal{L}_{\text{reg.}}$ denotes the weighted sum of additional regularization terms: $\mathcal{L}_{\text{smooth}}$, $\mathcal{L}_{\text{Eikonal}}$, $\mathcal{L}_{\text{skymask}}$.

We follow prior works [25, 28] and regularize the gradient

of SDF value s with an Eikonal term:

$$\mathcal{L}_{\text{Eikonal}} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} (||\nabla_{\mathbf{x}} s(\mathbf{x})||_2 - 1)^2, \tag{2}$$

where $\mathcal{X}$ is the set of points sampled along the ray.

Similar to prior inverse rendering works [1, 7, 12, 17], we also encourage local smoothness of normals and material properties. Specifically, we follow Nvdiffrecmc [7] and apply the smoothness regularization for base color $\mathbf{k}_d$, normal $\mathbf{n}$, and material $\mathbf{k}_s$:

$$\begin{aligned} \mathcal{L}_{\text{smooth}} =& \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} |\mathbf{k}_d(\mathbf{x}) - \mathbf{k}_d(\mathbf{x} + \epsilon)| \\ &+ \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} |\mathbf{k}_s(\mathbf{x}) - \mathbf{k}_s(\mathbf{x} + \epsilon)| \\ &+ \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} |\mathbf{n}(\mathbf{x}) - \mathbf{n}(\mathbf{x} + \epsilon)|, \end{aligned} \tag{3}$$

where $\epsilon \sim \mathcal{N}(0, \sigma = 0.02)$ is a local perturbation vector.

Finally, prior works such as NeRF-OSR [21] do not explicitly handle the sky and hence produce many floaters in their scene representation. In our work, we follow [26] and apply a binary cross entropy (BCE) loss $\mathcal{L}_{\text{skymask}}$ between the volume rendered alpha channel and the sky semantic segmentation masks. The sky masks are again obtained from an off-the-shelf semantic segmentation network [23]. In practice, we assign a small weight to this sky mask regularization to only carve out the floaters in the sky region, while not harming the geometry of the scene.

Training details. The final loss is a weighted sum of the reconstruction and regularization terms

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{\text{render}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}} + \lambda_{\text{rad.}} \mathcal{L}_{\text{rad.}} \\ &+ \lambda_{\text{norm.}} \mathcal{L}_{\text{norm.}} + \lambda_{\text{shade}} \mathcal{L}_{\text{shade}} \\ &+ \lambda_{\text{Eikonal}} \mathcal{L}_{\text{Eikonal}} + \lambda_{\text{skymask}} \mathcal{L}_{\text{skymask}} \\ &+ \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}}. \end{aligned} \tag{4}$$

where the weight of each loss function is set to: $\lambda_{\text{rad.}} = \lambda_{\text{norm.}} = 1$, $\lambda_{\text{shade}} = 0.1$, $\lambda_{\text{Eikonal}} = 0.05$, $\lambda_{\text{skymask}} = \lambda_{\text{smooth}} = 0.01$. λ_{depth} is set to 1 on Driving data and 0 for NeRF-OSR dataset [21]. We use Adam optimizer [10] with a learning rate of 1e-2. As the mesh extraction requires a well initialized SD field, we run a warm-up phase for 5k iterations in which we remove $\mathcal{L}_{\text{render}}$ and $\mathcal{L}_{\text{shade}}$. After the warm-up phase we continue optimizing all the loss terms for additional 50k iterations. In each batch, we sample 4096 rays. With the parameters detailed above, FEGR consumes about 20GB GPU memory during training.

C. Experiment Analysis and Results

In this section, we provide a detailed experimental setup and additional results.

Relighting details. The application of *relighting* aims to generate imagery of the 3D scene under the lighting conditions specified by the users, typically an HDR environment map. FEGR represents the scene with standard PBR materials, and thus can directly replace the reconstructed HDR environment light $f_{\text{env.}}$ with the user-specified lighting.

The NeRF-OSR [21] baseline requires spherical harmonics lighting, and thus we converted the HDR map to an SH representation as suggested in the paper[1]. In the qualitative comparison (main paper and the accompanied video), we tackle a more challenging scenario and use a high-contrast HDR map with strong directional light to highlight the ability of the methods to cast shadows. In this case, NeRF-OSR shows relatively worse qualitative performance, with reasons in twofold: (i) The strong directional sunlight makes the small normal artifacts more pronounced, and (ii) The SH coefficients estimated from peaky HDR environment maps are not on the training data manifold, making the shadow network fail to generalize. In addition, NeRF-OSR implicitly represents shadows with an MLP learned across multiple illumination, and thus cannot guarantee that the shadows follow the rule of light transport.

Compared to NeRF-OSR, FEGR supports rendering the physics-based shadow effects from the user-specified lighting via ray-tracing, such as shadows due to self-occlusion. We refer to the accompanied video for qualitative comparison and additional results on relighting.

Object insertion details. The application of *virtual object insertion* takes as input synthetic objects with know geometry and materials, and aims to produce photorealistic imagery by placing them into real-world images. This requires proper handling of lighting effects such as cast shadows and specular highlights. For this image editing task, we follow the object insertion formulation in [26], which first separately renders the foreground objects and scene shadows, and then composite them onto the input scene image. The rendering is performed in Blender [6].

Existing works on inverse rendering [3, 4, 7, 17, 21, 29–32] typically adopt simplified lighting representations such as a point light [2, 22] or low-frequency spherical lobes [4, 21]. These works do not aim to estimate spatially-varying lighting. Instead, they only use lighting as a side-product in the joint optimization process and they discarded it after training.

We compare FEGR on the task of virtual object insertion with recent state-of-the-art learning-based outdoor lighting estimation methods [8, 26]. Qualitative comparison is available in main paper Fig. 6 and a user study in main paper Table 3. For the user study, we follow the setup of [26] and conduct it on Amazon Mechanical Turk. Compared to learning-based feed-forward lighting estimation models, we acknowledge that our method consumes more information as

[1] We use this repository to estimate the SH coefficients

Training image Ours Ours (w/o $\mathcal{L}_{\text{shade}}$)

Figure B. **Qualitative ablation of shading prior.** We qualitatively ablate the effect of the semantic-aware shading regularization loss $\mathcal{L}_{\text{shade}}$. For each scene, we visualize the estimated HDR environment map and an object insertion result. On the bottom-right of the environment map, we divide the HDR value by 30 to better display the HDR component of the environment map.

Figure C. Qualitative visualization of mesh reconstruction. We visualize the underlying geometry reconstructed by our method.

input and requires online optimization. However, we stress that our method achieves significantly improved results and recovers accurate shadow direction and intensity, which is challenging for single-image feed-forward methods. We believe that our formulation can inspire future works on the role of lighting in optimization-based inverse rendering.

We refer to the accompanied video for additional results on virtual object insertion.

Qualitative ablation of shading prior $\mathcal{L}_{\text{shade}}$. We qualitatively ablate and show the results in Fig. B. When training without the shading prior loss term $\mathcal{L}_{\text{shade}}$, the estimated environment light can still predict the peak direction but typically fails to produce sharp cast shadows and correct shadow scale. This indicates the shading prior $\mathcal{L}_{\text{shade}}$ is beneficial for HDR light estimation.

Qualitative visualization of meshes. In Fig. C, we visualize the underlying geometry extracted by marching cubes. In the hybrid rendering described in main paper Sec. 3.2, the mesh accounts for the visibility query of secondary rays to render cast shadows.

References

[1] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. IEEE transactions on pattern analysis and machine intelligence, 37(8):1670–1687, 2014. 2, 3

[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 preprint arXiv:2008.03824, 2020. 3

[3] 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. In ECCV, pages 294–311. Springer, 2020. 3

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In ICCV, 2021. 2, 3

[5] Brent Burley. Physically-based shading at disney. 2012. 1

[6] Blender Online Community. Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 3

[7] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. arXiv:2206.03380, 2022. 1, 2, 3

[8] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In CVPR, pages 6927–6935, 2019. 3

[9] Brian Karis and Epic Games. Real shading in unreal engine 4. Proc. Physically Based Shading Theory Practice, 4(3), 2013. 1

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[11] Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. NeROIC: Neural object capture and rendering from online image collections. Computing Research Repository (CoRR), abs/2201.02533, 2022. 1

[12] Edwin H Land and John J McCann. Lightness and retinex theory. Josa, 61(1):1–11, 1971. 3

[13] 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 CVPR, pages 2475–2484, 2020. 2

[14] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput. Graph., 21(4):163–169, aug 1987. 2

[15] 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. arXiv preprint arXiv:2003.08934, 2020. 2

[16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (TOG), 41(4):102:1–102:15, July 2022. 2

[17] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. arXiv:2111.12503, 2021. 3

[18] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. ACM Trans. Graph., 29(4), jul 2010. 2

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32, 2019. 2

[20] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. CVPR, 2022. 1

[21] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In ECCV, 2022. 3

[22] 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. 3

[23] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. arXiv preprint arXiv:2005.10821, 2020. 2, 3

[24] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. CVPR, 2022. 1

[25] 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. 1, 2

[26] Zian Wang, Wenzheng Chen, David Acuna, Jan Kautz, and Sanja Fidler. Neural light field estimation for street scenes with differentiable virtual object insertion. In ECCV, 2022. 3

[27] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3d spatially-varying lighting. In ICCV, 2021. 2

[28] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In Thirty-Fifth Conference on Neural Information Processing Systems, 2021. 2

[29] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images. In CVPR, 2022. 1, 3

[30] 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. 3

[31] 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 (TOG), 40(6):1–18, 2021. 3

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