MILO: Multi-bounce Inverse Rendering for Indoor Scene with Light-emitting Objects

Bohan Yu, Siqi Yang, Xuanning Cui, Siyan Dong, Baoquan Chen, *Fellow, IEEE*
Boxin Shi, *Senior Member, IEEE*

Abstract—Recently, many advances in inverse rendering are achieved by high-dimensional lighting representations and differentiable rendering. However, multi-bounce lighting effects can hardly be handled correctly in scene editing using high-dimensional lighting representations, and light source model deviation and ambiguities exist in differentiable rendering methods. These problems limit the applications of inverse rendering. In this paper, we present a multi-bounce inverse rendering method based on Monte Carlo path tracing, to enable correct complex multi-bounce lighting effects rendering in scene editing. We propose a novel light source model that is more suitable for light source editing in indoor scenes, and design a specific neural network with corresponding disambiguation constraints to alleviate ambiguities during the inverse rendering. We evaluate our method on both synthetic and real indoor scenes through virtual object insertion, material editing, relighting tasks, and so on. The results demonstrate that our method achieves better photo-realistic quality.

Index Terms—Inverse Rendering, Intrinsic Image Decomposition, Multi-bounce.

1 Introduction

Inverse rendering is desired in many computer vision and graphics applications, such as novel view synthesis, virtual object insertion, material editing, relighting, and so on. With light source and material information recovered through inverse rendering, these applications can be accomplished by re-rendering with photorealistic quality. Traditional intrinsic image decomposition methods formulate the inverse rendering problem as decomposing photographs into reflectance and shading maps [3], [40], [49]. Modern lighting estimation methods represent incident light by spherical harmonics [55], spherical Gaussian [25] models, multi-layer perceptron [30], or voxel-like structures [41], [48]. General Bidirectional Reflectance Distribution Function (BRDF) models are also introduced to inverse rendering to represent more realistic reflectance [1], [11], [17].

However, inverse rendering approaches via intrinsic image decomposition and lighting estimation have a common limitation: Only the last bounce reflection is calculated in rendering, while multi-bounce lighting effects are represented by the incident lighting models. Applications that require post-editing the reconstructed scene also change lighting, *e.g.*, lighting is changed directly in relighting tasks and indirectly in virtual object insertion or material editing tasks. These changes can hardly be applied to high-dimensional lighting representations. Existing approaches [30], [42] usually implement these tasks by ignoring complex cast shadow, virtual light illumination, and some multi-bounce effects, which result in unrealistic post-edited images.

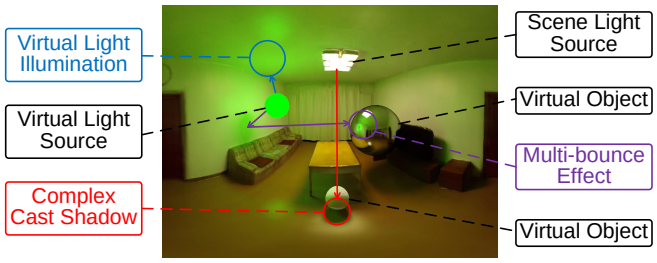


Fig. 1. Three virtual balls (light-emitting, specular, and diffuse) are inserted into a real indoor scene. MILO renders photo-realistic images by reasonably demonstrating complex cast shadows, multi-bounce effect between scene objects and virtual objects, and virtual light illumination. For example, the specular reflection of the ceiling light is partially occluded by the diffuse ball. The green light source also illuminates the walls and the ceiling to green, which is reflected by the specular ball.

Physically-based Monte Carlo path tracing [22], [24], [32], [54] is able to take multi-bounce into consideration and simulate light transport more precisely. However, applying path tracing to inversely rendering light sources and materials from images is a challenging task. Challenge *i)*: Inverse rendering requires a light source model that is able to describe physically light-emitting objects (instead of an intermediate lighting representation), the model should be directly visible. However, existing light source models, such as point light, directional light, spotlight, and environment map, do not satisfy these requirements; Challenge

- *B. Yu and B. Shi are with the National Engineering Research Center of Visual Technology, School of Computer Science, Peking University, Beijing 100871, China.*
  E-mails: ybh1998@pku.edu.cn, shiboxin@pku.edu.cn.
- *S. Yang is with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China.*
  E-mail: yousiki@pku.edu.cn.
- *X. Cui is with the AI Innovation Center, School of Computer Science, Peking University, Beijing 100871, China.*
  E-mail: cxn@pku.edu.cn.
- *S. Dong is with the Interdisciplinary Research Center, Shandong University, Shandong 250100, China.*
  E-mail: siyandong.3@gmail.com.
- *B. Chen is with the School of Intelligence Science and Technology, Peking University, Beijing 100871, China.*
  E-mail: baoquan@pku.edu.cn.
- *Corresponding author: B. Shi.*

ii): Path tracing introduces a larger number of unknown variables than single-bounce methods, which cause three ambiguities in inverse rendering: 1) multi-bounce ambiguity (each bounce has a different surface material), 2) emission-diffuse ambiguity (direct emission and reflection are hardly distinguishable from appearance), and 3) specular uncertainty (there are regions in which specular reflection is not observed in any of the input images). Due to these ambiguities, different combinations of light sources and materials can produce the same appearance and lead to errors in re-rendering edited scenes.

To conquer the above two challenges, we propose **MILO**: **M**ulti-bounce **I**nverse rendering for indoor scene with **L**ight-emitting **O**bjects, consisting of **MILO-Renderer** and **MILO-Net**. Our method is based on modern Monte Carlo path tracing to render complex multi-bounce lighting effects. For the light source, we propose a novel physically-based light-emitting object model which represents light-emitting objects inside the room, such as ceiling lights, lamps, and windows that pass light coming from the outside world. We implement this light source model in our differentiable renderer named MILO-Renderer. To alleviate the ambiguities during inverse rendering, we propose three critical disambiguation constraints, *i.e.*, 1) sufficient sampling constraint, 2) finite material constraint, and 3) reflectance similarity constraint. We use these constraints to design a specific neural network named MILO-Net to predict light source and material parameters. Thanks to these designs, MILO applies complex light transport to the edited scenes and renders more photo-realistic images on post-edited scenes as shown in Figure 1.

MILO is a demonstration of scene-level inverse rendering that recovers **fully** spatially varying light sources and materials by leveraging differentiable path tracing. MILO is able to capture complex interactions between virtual objects and scene objects in virtual object insertion tasks. This is because MILO recovers physically light-emitting objects and simulates multi-bounce light transmission, which is not supported in existing inverse rendering methods [6], [20], [25], [30], [42], [55]. To sum up, our contributions are:

- We propose a novel light-emitting object model, which can represent common light sources (*e.g.*, ceiling lights and windows) in indoor scenes. This model is more suitable for scene editing, and friendly to differentiable rendering.
- We design a specific MILO-Net containing disambiguation constraints, which alleviates major errors in recovered light sources and materials caused by ambiguities in path tracing based differentiable rendering.
- We implement a Monte Carlo path tracing differentiable renderer — MILO-Renderer, which renders multi-bounce lighting and achieves superior rendering results with better photo-realistic quality.

2 Related Work

Lighting and reflectance information can be inversely estimated from images via intrinsic image decomposition, lighting estimation, inverse light transport, and differentiable rendering. In this section, we briefly review related works in these categories.

Intrinsic image decomposition methods aim at decomposing an image into a pixel-wise multiplication of several photometric components, which usually contain diffuse albedo and shading (including the interaction between lighting and surface normal). Recent works use more realistic microfacet BRDF models [1], [11], [17] for reflectance. We recommend the readers refer to related survey papers [8], [28] for more details.

Lighting estimation methods usually use high-dimensional data structures to represent lighting in the scene, such as SVSG (Spatially Varying Spherical Gaussian) [25], SVSH (Spatially Varying Spherical Harmonics) models [4], [38], [55], MLP (Multilayer Perceptron) [30], and voxel-like structures [41], [48]. Recent works recover reflectance and lighting together and achieve promising inverse rendering results [5], [10], [25], [48]. However, lighting

TABLE 1
Comparison among recent works on indoor scene inverse rendering using intrinsic image decomposition (IID) [25], [55], lighting estimation (LE) [30], [42], inverse light transport (ILT) [6], [20], [50], and differentiable rendering (DR) [2], [31] (including ours) methods. Inputs are abbreviated as S. Img. (Single Image), M. Img. (Multi-view Image), Geo. (Geometry), and Seg. (Object segmentation). Reflectance models are abbreviated as P-SVBRDF (Partially Spatially Varying BRDF) and F-SVBRDF (Fully Spatially Varying BRDF).

Method	Type	Input	Lighting	Reflectance	Detail Level	Multi-bounce Reflection
Zhou *et al.* [55]	IID & LE	S. Img.	SVSH	Lambertian	Pixel	Included in Lighting Model
Li *et al.* [25]			SVSG	F-SVBRDF		
Wang *et al.* [48]			Voxel-like			
Mildenhall *et al.* [30]	LE	M. Img.	MLP	N/A	Continuous	
Srinivasan *et al.* [42]			Voxel-like		Voxel	
Bi *et al.* [6]	ILT		PL	F-SVBRDF		No
Kim *et al.* [20]		Geo. & M. Img.		Lambertian	Pixel	
Yu *et al.* [50]				P-SVBRDF*	Mixed†	Approximated
Azinović *et al.* [2]	DR	Geo. & M. Img. & Seg.	LO (Object)		Object	Path Tracing
David *et al.* [31]					Mixed‡	
MILO (Ours)		Geo. & M. Img.	LO (Continuous)	F-SVBRDF	Continuous	

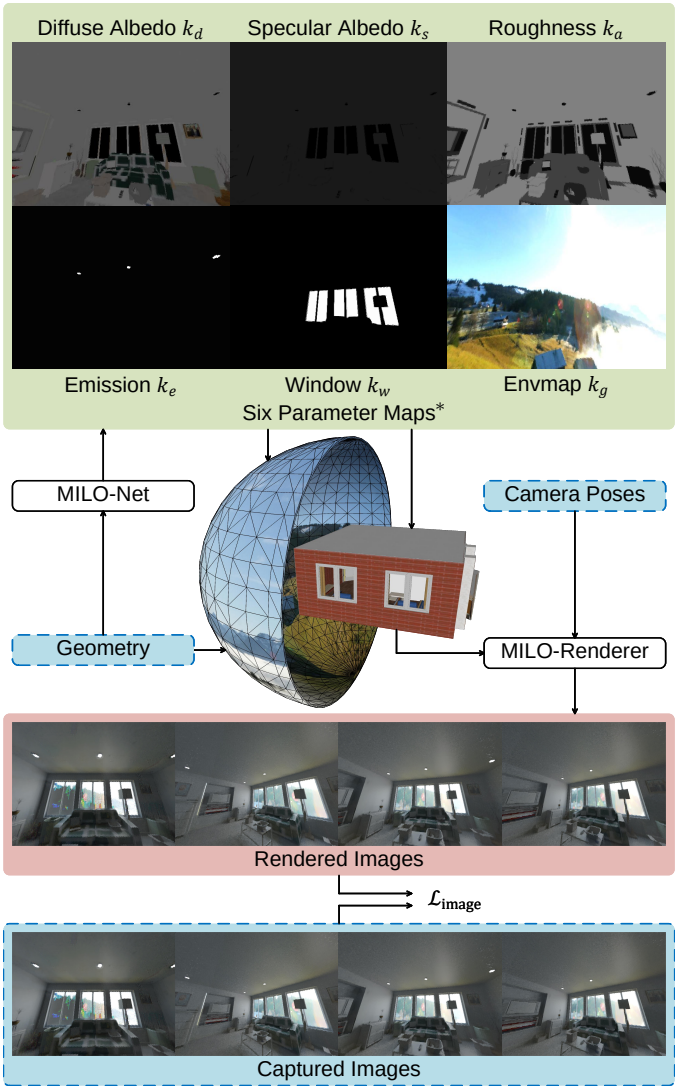
∗. These methods recover BRDF with parameters shared between different 3D points. We call these methods P-SVBRDF (Partially Spatially Varying BRDF). In contrast, methods recovering BRDF independently at different points are called F-SVBRDF (Fully Spatially Varying BRDF).
†. Specular related parameters in BRDF are shared within each surface. Diffused parameters are at pixel detail level.
‡. Specular related parameters in BRDF are shared within each object. Diffused parameters are at pixel detail level.

contains both direct emission from light sources and indirect reflection from other objects, as a result, it is difficult to edit the recovered lighting.

Inverse light transport methods recover light sources and reflectance parameters in the 3D space. Different from lighting models, light source models only describe direct light emission. Commonly used light source models are: point light (PL) sources [18], [20], [50], spotlights [52], light-emitting objects [2], [18], and environment maps. Inverse light transport can discriminate direct emission and reflection for achieving good scene editing results, but these simplified light source models can hardly describe complex light sources in the real world.

Differentiable rendering methods calculate derivatives of the rendered image with respect to scene parameters, which include rasterization based methods [19], [26], [27], path tracing based methods [24], [32], [54], and voxel ray marching based methods [6]. These methods mainly focus on calculating derivatives of step functions that commonly exist in rendering due to geometry discontinuity [54]. Differentiable rendering methods are also used in inverse rendering based on learning and optimization. This category of method takes advantage of the more accurate light transport simulation. It recovers light source and BRDF parameters simultaneously. However, optimizing a large number of independent parameters in the whole scene simultaneously is still difficult due to the ambiguities. Methods by Azinović *et al*. [2] and David *et al*. [31] reduce the number of parameters by object segmentation. They recover all [2] or part [31] of the reflectance and light source properties at the object level. Methods by David *et al*. [31] and Yu *et al*. [50] recover diffuse albedo of the reflectance parameters at the pixel level to enrich details, but other parameters are still shared at the object level. These methods share some of the BRDF parameters at different 3D points, which we name as partially spatially varying BRDF. It is still challenging to recover reflectance parameters at the fully spatially varying BRDF level.

In this paper, we focus on recovering reflectance and light source parameters using path tracing based differentiable rendering. We circumvent the step function derivative problem by proposing a novel light source model and alleviate ambiguities by a specially designed neural network. The comparison between some related inverse rendering works and MILO (ours) is listed in Table 1.



∗. The parameter maps displayed here are from the ground truth of this scene for better illustration. The recovered environment map k_g only contains part of the outdoor scenes visible through the window.

Fig. 2. MILO pipeline. MILO-Net takes scene geometry as input to predict light-emitting and reflectance properties for differentiable path tracing. MILO-Renderer uses scene geometry, six parameter maps, and camera poses to render images. Rendered images are compared with captured images.

3 Method

MILO consists of two parts: MILO-Renderer and MILO-Net. The complete pipeline is shown in Figure 2. MILO takes the scene geometry and a set of High Dynamic Range (HDR) images captured from different viewpoints with their camera poses as input. MILO-Net predicts six parameter maps from geometry to describe the properties of light-emitting objects and the reflectance of the scene. Three of them describe material properties: diffuse albedo map k_d, specular albedo map k_s, and roughness map k_a. The other three describe light-emitting object properties: emission map k_e, window map k_w, and outdoor environment map k_g. MILO-Renderer conducts inverse path tracing, which uses six parameter maps, scene geometry, and camera poses to render images and calculates the derivatives of parameter maps in back propagation.

3.1 Problem Formulation and Light Source Model

Given the geometry and a set of images of an indoor scene, each pixel value is converted to an observed radiance, which is denoted as $\tilde{L}_j, j = \{1, 2, \cdots, M\}$. The total number of observed radiance values M equals the number of images times the number of pixels per image. The reflected radiance $L(\boldsymbol{\omega_r}, \boldsymbol{x})$ at scene surface point $\boldsymbol{x}$ with reflected light direction $\boldsymbol{\omega_r}$ can be formulated by the rendering equation [16] as

$$L(\boldsymbol{\omega_r}, \boldsymbol{x}) = E(\boldsymbol{\omega_r}, \boldsymbol{x}) + \int_{\Omega} f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}, \boldsymbol{x}) L(\boldsymbol{\omega_i}, \boldsymbol{y}) \cos\theta \mathrm{d}\boldsymbol{\omega_i}. \quad (1)$$

E is the light source model and f_r is the BRDF model. $L(\boldsymbol{\omega_i}, \boldsymbol{y})$ is incident radiance from angle $\boldsymbol{\omega_i}$. $\boldsymbol{y}$ is the source point of the incident ray, which is calculated by intersecting incident ray with the scene. θ is the angle between surface normal and $\boldsymbol{\omega_i}$. We assume that radiance does not decay as light transports in the air. The incident radiance is integrated over the upper hemisphere Ω.

To model light-emitting objects such as indoor lights and windows in the scene, we introduce three parameter maps: the emission map k_e (RGB), the window map k_w in range $(0, 1)$, and the outdoor environment map k_g (RGB) to represent E as

$$E(\boldsymbol{\omega}, \boldsymbol{x}) = k_e(\boldsymbol{x}) + k_w(\boldsymbol{x}) \cdot k_g(\boldsymbol{\omega}). \tag{2}$$

In our light source model, all light sources adhere to the object surface and are directly visible from the camera. The emission map k_e describes the radiance of objects that emit omnidirectional light. The window map k_w describes the capacity that a window surface passes light from the outside world. And the outdoor environment map k_g describes the radiance of outdoor lighting from incident angle $\boldsymbol{\omega}$. Indoor lights such as ceiling lights and lamps are represented by k_e. We assume that these light sources emit omnidirectional light, which cast identical radiance in every direction on the upper hemisphere of the surface, so the radiance is a function only w.r.t. light position $\boldsymbol{x}$. On the other hand, the light passing through the window is represented by k_w and k_g. We assume that windows are fully transparent, so k_w is a binary function w.r.t. window position $\boldsymbol{x}$. We assume that the sun, sky, and other outdoor scenes are far enough so that k_g is a function only w.r.t. ray direction $\boldsymbol{\omega}$. Different from traditional light source representations such as point light and area light, we use continuous functions to describe light source properties. The position $\boldsymbol{x}$ and direction $\boldsymbol{\omega}$ are fixed while the radiance functions are being optimized. In this way, we avoid suffering from step function gradient problem for cast shadows and alleviate local minima problems caused by the light source position.

To represent the reflectance properties of the scene, we use the Cook-Torrance BRDF model [11] as f_r, which is a function of the diffuse albedo map k_d (RGB), specular albedo map k_s(RGB), and roughness map k_a in range $(0, 1)$:

$$\begin{aligned} f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}, \boldsymbol{x}) &= \frac{1}{\pi} k_d(\boldsymbol{x}) + \frac{DFG}{\pi(\boldsymbol{\omega_i} \cdot \boldsymbol{N})(\boldsymbol{N} \cdot \boldsymbol{\omega_r})} \\ D &= \frac{1}{\pi k_\alpha (\boldsymbol{N} \cdot \boldsymbol{H})^4} \cdot \exp\left(\frac{(\boldsymbol{N} \cdot \boldsymbol{H})^2 - 1}{k_\alpha (\boldsymbol{N} \cdot \boldsymbol{H})^2}\right) \\ G &= \min\left(1, \frac{2(\boldsymbol{H} \cdot \boldsymbol{N})(\boldsymbol{\omega_i} \cdot \boldsymbol{N})}{\boldsymbol{\omega_i} \cdot \boldsymbol{H}}, \frac{2(\boldsymbol{H} \cdot \boldsymbol{N})(\boldsymbol{\omega_r} \cdot \boldsymbol{N})}{\boldsymbol{\omega_i} \cdot \boldsymbol{H}}\right) \\ F &= k_s + (1 - k_s)(1 - \boldsymbol{N} \cdot w_i)^5. \end{aligned} \tag{3}$$

3.2 MILO-Renderer

MILO-Renderer calculates the radiance of a ray in a differentiable manner. Given the starting point and direction of a ray, Monte Carlo path tracing approximates the radiance by random walk:

$$L(\boldsymbol{\omega}, \boldsymbol{x}) = \frac{1}{S} \sum_{s=1}^{S} \sum_{b=0}^{B} E(\boldsymbol{\omega_{b,r}}, \boldsymbol{x_b}) \prod_{k=0}^{b-1} \frac{f_r(\boldsymbol{\omega_{k,i}}, \boldsymbol{\omega_{k,r}}, \boldsymbol{x_k})}{P(\boldsymbol{\omega_{k,i}}, \boldsymbol{\omega_{k,r}}, \boldsymbol{x_k})} \cos\theta. \tag{4}$$

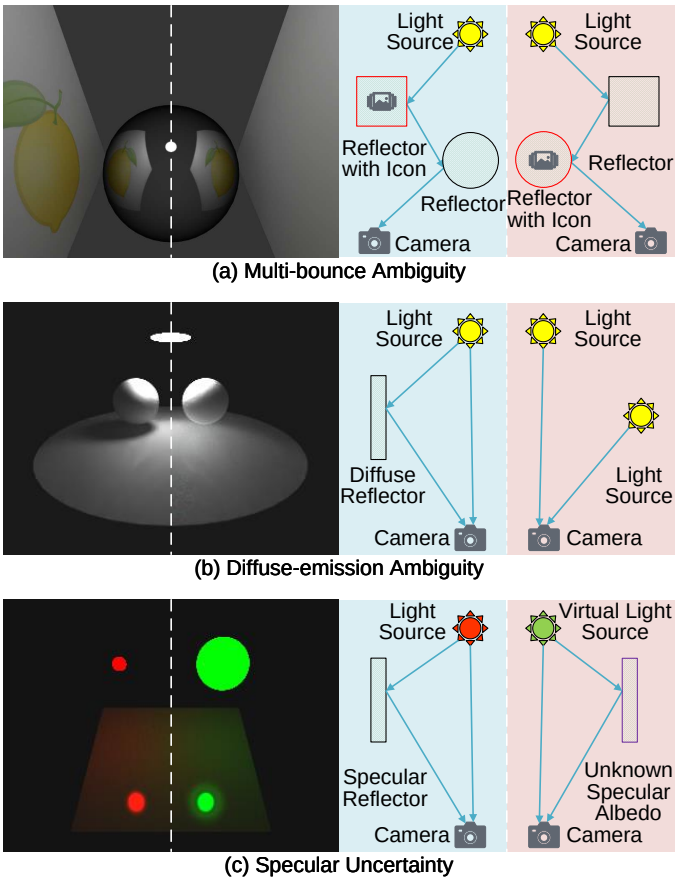


Fig. 3. Three types of ambiguities during inverse rendering: (a) Multi-bounce ambiguity: the icon on the wall (on the left side) and the icon on the specular ball (on the right side) both render the same result on the specular ball; (b) Diffuse-emission ambiguity: the lower plate reflects light on the left side, while emitting light on the right side, consequently, the virtual ball on the right side has wrong shading and no shadow. (c) Specular uncertainty: the specular parameters are recovered only by the left side, however, if replacing the light source to a larger ball, the specular reflection is rendered correctly only in the small highlighted area on the right side.

S is the sampling number. B is the number of maximum bounces. $\boldsymbol{x_b}$ is the point on surface of the b-th bounce. $\boldsymbol{\omega_{b,i}}$ and $\boldsymbol{\omega_{b,r}}$ are incident and reflected ray directions at the b-th bounce. P is Probability Density Function (PDF) used for importance sampling [23]. Please refer to the experiment section for detailed importance sampling implementation.

MILO-Renderer uses Equations (2), (3), and (4) as the rendering function, in which six parameter maps are independent variables. The gradients of these parameter maps are calculated by backpropagation. Here we do not calculate derivatives w.r.t. PDF P [51] because it is also used by importance sampling. With MILO-Renderer, we can render the radiance of rays and compare them with observed values, using the following loss function:

$$\mathcal{L}_{\text{image}} = \sum_{j=1}^{M} \left\| L(\boldsymbol{\omega_j}, \boldsymbol{x_j}) - \tilde{L}_j \right\|_2. \tag{5}$$

3.3 Ambiguities and Disambiguation Constraints

MILO-Renderer provides the gradients of six parameter maps for their optimization. However, directly optimizing

these parameter maps will not produce a unique result. In general, there are infinite combinations of six parameter maps to satisfy minimizing the image loss $\mathcal{L}_{\text{image}}$, within which only one combination is correct. Next, we analyze three types of ambiguities causing infinite solutions and introduce our disambiguation constraints to select the correct combination.

Multi-bounce ambiguity. The rendered radiance of each pixel is influenced by light sources and all surface materials in the light transmission path. As a result, many combinations of light-emitting objects and materials could render the same observed radiance value for a pixel. Such an ambiguity exists between shading map and reflectance map in intrinsic image decomposition methods [3]. Multi-bounce ambiguity also exists between two materials at a higher number of bounces. An example is shown in Figure 3 (a): A specular ball in the middle is reflecting an icon on the left wall, however, the inverse rendering may misunderstand that the icon belongs to the ball's specular albedo while the wall is pure white, which renders the same result on the right side of the ball.

Diffuse-emission ambiguity. Diffuse reflection and emission are described by diffuse albedo map k_d and emission map k_e respectively. They both emit omnidirectional light, so we cannot distinguish them just by appearance. However, the diffuse reflection will disappear if the incident light is turned off or occluded by other objects while emission still exists, so it is important to recover them correctly in relighting and virtual object insertion tasks. As shown in Figure 3 (b), on the right side, the inverse rendering misunderstands that the lower plate is emitting light itself. Although the shading on the plate is the same as the left side, the inserted ball has no shadow and the shading is incorrect.

Specular uncertainty. To recover specular albedo k_s and roughness k_a, effective specular reflection observations and incident light that contains enough high-frequency information are required. The direct specular reflection of light sources satisfies these requirements, however, it is only visible in small highlighted regions. We can hardly cover all scene regions with only a few input images, thus the constraints for solving k_s and k_a in the uncovered area are deficient. As shown in Figure 3 (c), the specular albedo is only recovered correctly within the highlighted region while the remaining area is not well constrained. As a result, the edited light source renders incorrect specular reflection.

To solve these ambiguities, we propose three disambiguation constraints correspondingly:

- **Sufficient sampling constraint**: Every point in the scene should appear at least three times in the images with different observed radiance values.
- **Finite material constraint**: The whole scene is assumed consisting of only a finite number of different materials, including light sources and windows.
- **Reflectance similarity constraint**: The specular albedo and roughness at undetermined areas are supposed to have similar values to nearby regions or regions with similar appearance.

To be specific, we assume that the total number of different materials is Q (provided by users), within which only one light-emitting object material is used for windows and S light-emitting object materials are used for lights with learnable radiance. We ignore diffuse or specular reflections from light-emitting object materials since their radiance values are much larger than reflections.

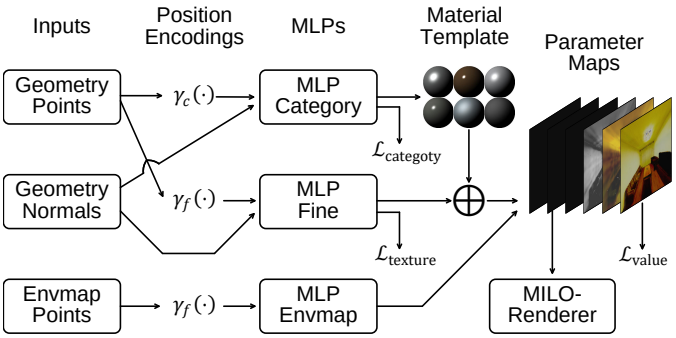


Fig. 4. MILO-Net takes geometry points, geometry normal, and environment map (Envmap) points as inputs to predict all six fully spatially-varying parameter maps.

To resolve multi-bounce ambiguity, the number of parameters of all materials should be less than or equal to the total number of constraints when minimizing $\mathcal{L}_{\text{image}}$. Each region on the surface should be either light-emitting object or non-light-emitting material. For light-emitting object, there are 3 unknown parameters to be solved (RGB values of either k_e or k_g), while for each non-light-emitting material, there are 7 unknown parameters to be solved (RGB values of k_d, RGB values of k_s, and k_a). Each observed ray provides 3 constraints (RGB values of the point). Since the sufficient sampling constraint requires each point to appear at least 3 times, there are at least 9 constraints for any region in the scene, so the total number of constraints is greater than unknown parameters to make Equation (1) well constrained. However, sometimes the difference in radiance is too small to be captured by cameras, *e.g.*, the material has too low specular albedo (Lambertian) or too low roughness, or the lighting from different angles is similar. We can only alleviate multi-bounce ambiguity in these cases rather than eliminate it. In practice, to satisfy the sufficient sampling constraint during data acquisition, the photographer should capture several wide-baseline images that cover the region as much as possible.

To resolve diffuse-emission ambiguity, we apply the finite material constraint. As shown in Figure 3 (b), the emission map of the object should be a gradient map to fit the rendered image, which cannot be represented by a finite number of materials. For the conservation of energy, the brightest area (the upper plate) should belong to the light-emitting object. With the finite material constraint, we can easily determine several material parameters with sufficiently observed radiance values from images.

To resolve specular uncertainty, the reflectance similarity constraint and finite material constraint are used together. These constraints are implemented in MILO-Net, a neural network that predicts parameter maps from geometry.

3.4 MILO-Net

We propose a neural network named MILO-Net to predict material maps from the given geometry. It is designed to

fulfill the following three functionalities: *i)* To generate fully spatially varying light source and reflectance parameter maps that have independent values at different 3D points; *ii)* to implement finite material constraint, and *iii)* reflectance similarity constraint as mentioned above. The structure of MILO-Net is illustrated in Figure 4.

To generate fully spatially varying parameter maps, we use MLPs with position encoding [30] as smooth projection functions from 3D points to materials. For parameter maps attached to the geometry $(k_d, k_s, k_a, k_e, k_w)$, we use both geometry point positions and normals as input. For the outdoor environment map, we generate a unit sphere and use point positions on the sphere as input. The position encoding enriches fine grained details of the predicted result. The positions are first normalized to the range of $[-1, 1]$. Then we use position encoding $\gamma(\boldsymbol{x})$ to enhance quality of the predicted parameter maps. For a point $\boldsymbol{x}$ in the scene, the position encoding function is:

$$\begin{aligned}\gamma(\boldsymbol{x}) = \Big(&\sin\left(2^0\pi\boldsymbol{x}\right), \cos\left(2^0\pi\boldsymbol{x}\right), \\ &\sin\left(2^1\pi\boldsymbol{x}\right), \cos\left(2^1\pi\boldsymbol{x}\right), \\ &... \\ &\sin\left(2^{L-1}\pi\boldsymbol{x}\right), \cos\left(2^{L-1}\pi\boldsymbol{x}\right)\Big).\end{aligned} \tag{6}$$

At the end of the neural network, we add kernel functions to map the output to the proper range.

Finite material constraint requires that the scene consists of only Q different materials. By concatenating all parameter maps $(k_d, k_s, k_a, k_e, k_w)$ into an 11-D parameter map k_m, finite material constraint is formulated as

$$\tilde{k}_m(\boldsymbol{x}) = \sum_{i=1}^{Q} P_i(\boldsymbol{x}) \cdot \boldsymbol{m_i}, \tag{7}$$

where $\boldsymbol{m_i}$ is a material template with Q learnable materials, P_i is the probability that the current point belongs to the i-th material. The P_i values for each 3D point are predicted by the first MLP. We use softmax function to convert MLP outputs to material probability P_i and generate parameter maps $\tilde{k}_m$. To make sure each point belongs to only one material, we add the category loss $\mathcal{L}_{\text{category}}$:

$$\mathcal{L}_{\text{category}} = -\sum_{\boldsymbol{x}} \max_i P_i(\boldsymbol{x}). \tag{8}$$

Reflectance similarity constraint requires that two points should have similar parameter values if they are close in 3D space. In another word, the gradient of the generated parameter maps w.r.t. the inputs $\boldsymbol{x}$ should be small as possible. This gradient is a production of position encoding gradient and MLP gradient. Higher level of position encoding results in a larger gradient, so we configure the smoothness of material maps by tuning the level of position encoding $\gamma_c(\boldsymbol{x})$ [45]. To reduce the gradient of MLP, we also add ℓ_2 regularization term to the first MLP.

The material template ensures finite material constraint, however, it makes 3D points of the same material consistent with each other. The lower level of position encoding and ℓ_2 regularization term ensure reflectance similarity constraint, but they make the generated parameter maps over-smoothed. In order to keep the parameter maps fully spatially varying and rich in high-frequency details, we introduce detailed texture maps k_t. This map is generated using the second MLP with higher level of position encoding $\gamma_f(\boldsymbol{x})$, and added to $\tilde{k}_m$. We use ℓ_1 regularization term on k_t to make the network prefer generating parameter maps using the first MLP:

$$k_m(\boldsymbol{x}) = \tilde{k}_m(\boldsymbol{x}) + k_t(\boldsymbol{x}), \tag{9}$$

$$\mathcal{L}_{\text{texture}} = \sum_{\boldsymbol{x}} \left\| k_t(\boldsymbol{x}) \right\|_1. \tag{10}$$

In this way, the functionalities of the above two constraints are maintained. Finally, the outdoor environment map k_g is generated using the third MLP. Detailed structures of these three MLPs are shown in the experiment section.

To prevent producing invalid values (*e.g.*, negative albedo, negative emission, $k_d + k_s \geq 1.0$) or extreme values (*e.g.*, zero roughness) for parameter maps, which cause rendering error or gradient explosion, we add another value loss $\mathcal{L}_{\text{value}}$ to limit input values within a correct range before rendering. Finally, we render the predicted parameter maps using MILO-Renderer and optimize MILO-Net parameters to minimize the total loss:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{image}} + \beta \cdot \mathcal{L}_{\text{category}} + \theta \cdot \mathcal{L}_{\text{texture}} + \zeta \cdot \mathcal{L}_{\text{value}}. \tag{11}$$

For each scene, we train MILO-Net on the fly, which does not rely on a large amount of data for pre-training. All learnable network parameters in MILO-Net are optimized using AMSGrad [39] optimizer (for MLP parameters), and SGD optimizer with Nesterov momentum [44] (for material template parameters). After training, we predict six parameter maps and save them for post-editing. By modifying camera poses, geometry, and predicted parameter maps, we can re-render desired images.

3.5 Implementation

MILO-Net implementation. MILO-Net includes two position encodings, three MLPs (Multi-layer Perceptron), and a material template. MILO-Net is implemented using PyTorch [35]. The details are as follows:

For all three MLPs, we use 16-layer structures. The numbers of neurons at hidden layers are all 1024. We use LeakyReLU as the activation function, which uses 0.01 as the slope for negative values. We add skip connections (similar to residual networks) to the last 8 layers to solve the vanishing gradient problem. The level of position encoding L determines the smoothness of the predicted parameter maps. For lower-level position encoding γ_c, we use 10 as L to implement reflectance similarity constraint. For higher-level position encoding γ_f, we set L as 12 to enrich more high-frequency details. We use model parallelism to split the task on two graphics cards. Because training MLPs on all pixels requires too much VRAM, we only select 8.3% of the pixels stochastically for gradient calculation at each iteration.

For the material template, according to the sufficient sampling constraint, each material should be either light-emitting object material or normal material. For light-emitting object materials, diffuse albedo k_d and specular albedo k_s are set to 0, and roughness k_a is set to 0.05.

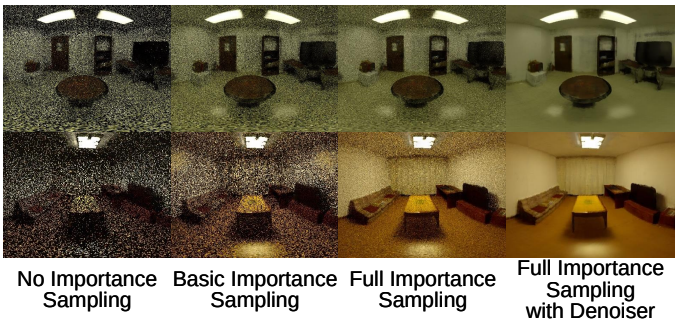


Fig. 5. Rendering results using different importance strategies. All images are rendered using 256 samples per pixel. Our full importance sampling strategy achieves the best result. We further reduce noise by denoiser for final result visualization.

Although the roughness map k_a doesn't affect the rendered image because specular albedo k_s is set to 0, it affects the process of optimization because other non-LO materials are soft combinations of items in the material template. We find that it is better to set a smaller value for k_a. For non-LO materials, emission k_e and window k_w are set to 0. Other parameters (k_e and k_w for light-emitting object material; k_d, k_s, and k_a for normal material) are learnable variables.

MILO-Renderer implementation. We implement MILO-Renderer in OpenCL [43]. There are two processes in MILO-Renderer: The forward process renders images from six parameter maps. The backward process calculates the derivatives of rendered images with respect to six parameter maps.

In the forward rendering process, MILO-Renderer renders the radiance value of a ray, given the scene geometry, parameter maps, start point, and direction of the ray. The geometry is stored in Wavefront OBJ format. In order to project each 3D point in the scene to a 2D coordinate on parameter maps, we unwrap the geometry using Blender. In order to calculate the ray-scene intersections efficiently, we build a Bounding Volume Hierarchy (BVH) acceleration structure, which splits the scene along three coordinates in turn and has a maximum of 20 levels. For each bounce, we use a mixed importance sampling strategy to sample incident ray direction effectively. The weights of the four sub-strategies are equally distributed:

- **Diffuse and specular reflection importance samplings**: We sample the incident ray direction using the modified Phong reflectance model [23]. This sampling strategy is commonly used in many renderers. The weights for diffuse and specular reflection importance samplings are both 0.25.
- **Environment map importance sampling**: We sample the incident ray direction with a probability that is proportional to the brightness of the predicted outdoor environment map. The weight for environment map importance sampling is 0.25.
- **Texture importance sampling**: We generate a Probability Density Function (PDF) map and sample points on the geometry as the incident ray direction. For light-emitting objects, this probability is proportional to the emission radiance value (average value of RGB). For normal materials, this probability is proportional to the image brightness (calculated by projecting camera-captured images to geometry). The weight for texture importance sampling is 0.25. For this strategy, calculating "all hits" during ray-scene intersections is required.

For each bounce, we select one of these strategies randomly according to their weights. The PDF of this mixed strategy is the weighted average of them. The maximum number of bounces B for each ray is 10, and the number of samples per ray S is 256. For better visualization, the final scene-editing results are further denoised by OpenImageDenoise [34]. This denoiser is not applied to the inverse rendering process.

To demonstrate the effectiveness of our mixed importance sampling method and denoiser, we render the image using no importance sampling, basic importance sampling (diffuse and specular reflection importance samplings only), full importance sampling (with all importance sampling strategies), and denoiser (using full importance sampling as the base image) in Figure 5 respectively. The results show that better importance sampling strategies reduce the sampling noise enormously.

In the backward rendering process, MILO-Renderer calculates the derivatives of rendered images with respect to six parameter maps. To prevent calculating ray-scene intersections twice, intermediate variables, such as "closest hit" of each ray-scene intersection and PDF of incident ray sampling, are saved during forward rendering. We can calculate the derivatives without calculating ray-scene intersections again in the backward process. To reduce memory consumption, only the last 3 bounces of each ray are saved

TABLE 2
Detailed information of 4 synthetic scenes and 5 real scenes. We evaluate MILO on different combinations of light source configurations.

Type	Name	Source	#Point Cloud Points	#Mesh Faces	#Images	#Lights	#LO Material	#Window Material	#Total Materials
Synthetic Scene	S1	AI2-Thor [21]	N/A	98,957	29	5	1	0	64
	S2			69,250	37	8	2	1*	64
	S3	PBRS [53]		51,986	30	0	1	1*	64
	S4			37,614	30	0	1	1*	64
Real Scene	R1	N/A	55,014,210	199,990	36	1	1	0	64
	R2		39,397,840	199,996	22	4	1	0	64
	R3		53,059,330	199,997	27	0	1	1	64
	R4		57,105,188	199,957	23	1	1	0	64
	R5		51,707,150	199,976	27	4	1	0	64

$*$. The ground truth outdoor environment maps are collected from Poly Haven [37]

and calculated in the backward process. Different rays may contribute gradient values to the same point on parameter maps, so we accumulate all gradient values by the "atomic float add" operation. When adding multiple gradient values to the same point simultaneously, the collision problem occurs, and the backward propagation efficiency is severely reduced. We duplicate the derivative parameter maps into 64 buckets, each ray selects the bucket according to its ray ID. In this way, we solve the collision problem and reduce backward time consumption greatly.

Optimization. At each iteration, we randomly sample $65,536$ rays from all images as a batch. The weights of all losses (α, β, θ, ζ) are set to 1.0, 0.003, 0.01, and 0.1 respectively. The weights for regularization terms (β, θ, ζ) are related to the number of samples S and parameter map resolutions, so they require readjustment for different settings accordingly. For each scene, the whole optimization procedure requires 2000 iterations to fully recover all six parameter maps. At each iteration, a batch of rays is sampled randomly. All rays are dynamically split and rendered on two graphics cards. It takes about 10 hours to train for one scene.

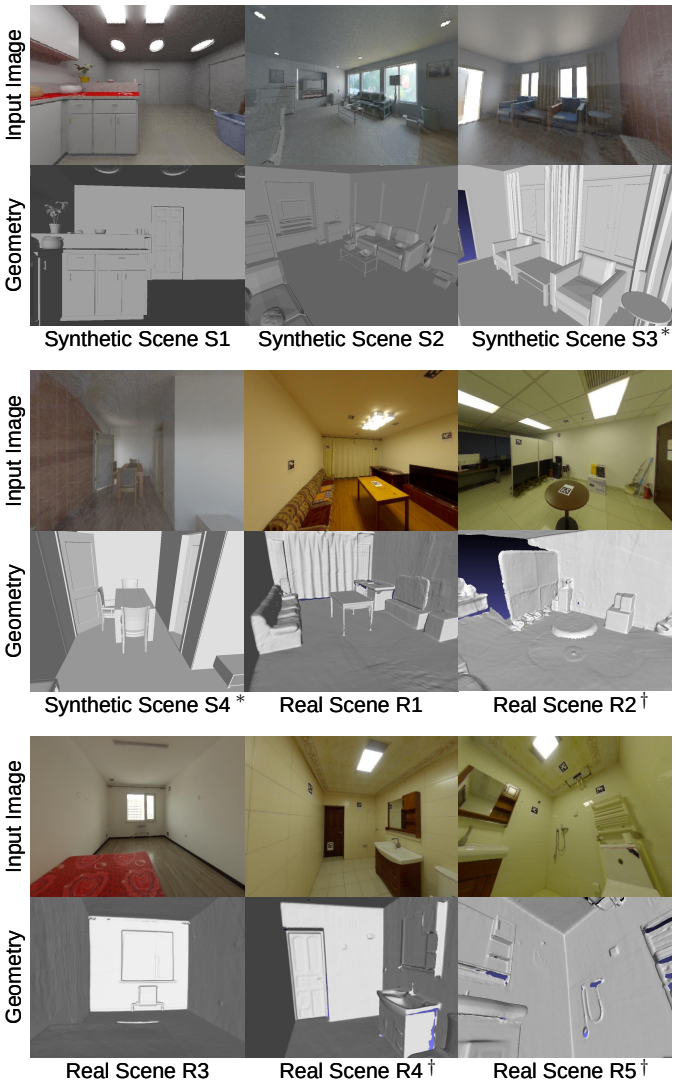
Because the variances of gradient values of material template parameters are related to the ray batch sampling and Monte Carlo sampling, there is high randomness in gradient variance. As a result, the adaptive gradient algorithms relying on gradient variances are not suitable. We choose the SGD optimizer with Nesterov momentum [44] to optimize the material template parameters. The initial learning rate is set to 0.01. Other MLP parameters are optimized using AMSGrad [39] with 0.001 initial learning rate. We decay the learning rates of the two optimizers by multiplying 0.3 at 400-th and 800-th iterations.

4 Experiment

Datasets. Our method is evaluated using 4 synthetic scenes and 5 real captured scenes. Two NVIDIA GeForce RTX 3090 graphics cards are used to render the dataset, train MILO-Net, differentiably render the scenes, and re-render post-edited scenes. We show input image examples, geometries, and detailed information of all synthetic and real scenes in Figure 6 and Table 2.

For synthetic scenes, we choose 2 scenes from the PBRS dataset [53] and 2 scenes from the AI2-THOR dataset [21]. The scenes in the PBRS dataset are originally rendered using the Mitsuba renderer [15], which is a CPU-based renderer. The scenes in AI2-THOR are originally rendered using Unity, which is not a physically-based renderer. For better performance and image reality, we convert these scenes to our format and render them using MILO-Renderer. We combine all objects in the scene together as a single Wavefront OBJ file, unwrap the geometry using Blender, and bake the textures into our six parameter maps representation. Several virtual cameras are placed into each scene manually to render images with light-source and material ground truth using MILO-Renderer.

For real scenes, we scan the geometries of 5 indoor scenes using the Intel RealSense D455 depth camera [14], and use Dot3D Pro software [12] to reconstruct and export dense point clouds with point normals from all the scenes.



∗. There are opened doors in these scenes. We use the outdoor environment map to represent light passing through the door.

†. There are holes in these real scenes due to the imperfect 3D scanning. For better visualization, we fill these holes using Poisson after rendering.

Fig. 6. Input image examples and geometries of 4 synthetic scenes and 5 real scenes. Our dataset covers different categories of rooms in indoor scenes.

Each scene contains about 50M points. We reconstruct triangular meshes using SSD (Smooth Signed Distance) surface reconstruction [9], and simplify the triangular meshes to about 20K triangular faces using MeshLab [29]. For capturing images, we place a RICOH THETA Z1 $360°$ camera [46] on a tripod to record HDR (High Dynamic Range) images using exposure bracketing. We choose a proper EV (Exposure Value) for each scene to capture light source radiance, then double the EV for 7 times. We stitch the two fisheye images using RICOH THETA Stitcher. To align geometry and captured images, we place several AprilTags [33] in the scene, which are detected and located by the Dot3D Pro software. After performing image alignment, we remove pixels with the tripod, AprilTags, and their cast shadows and specular reflections from input images. All six parameter maps

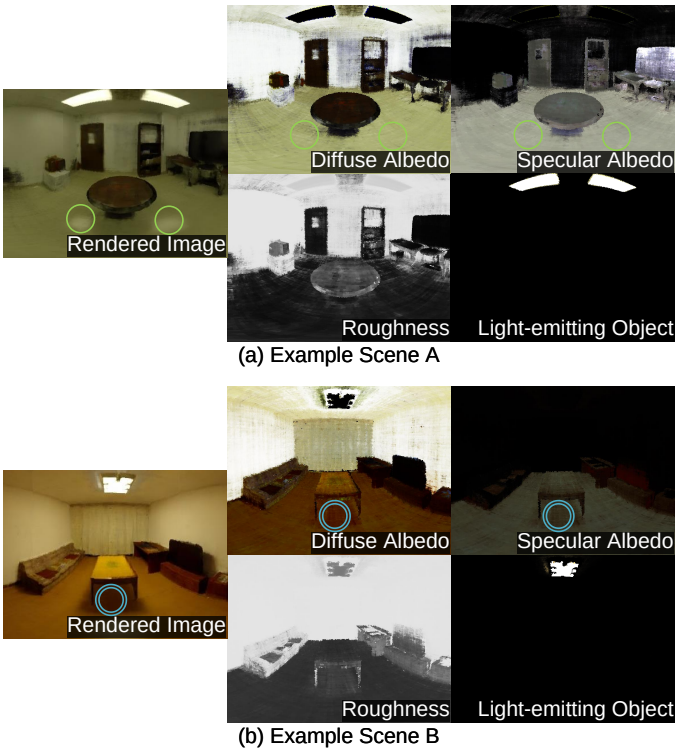


Fig. 7. Recovered parameter maps for two real scenes and the rendered images. The specular highlights (green circle) and cast shadows (blue double circle) are removed in diffuse and specular albedo. Only the ceiling lights are emitting light.

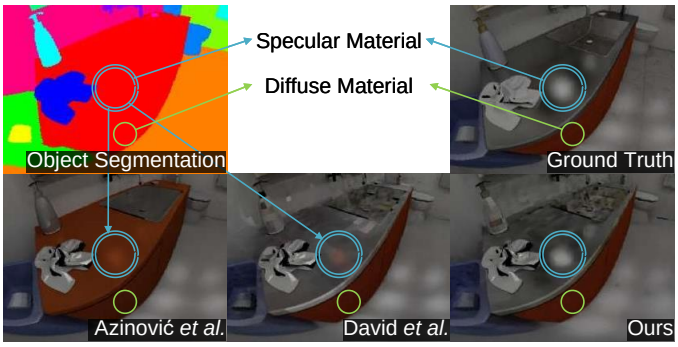


Fig. 8. Comparison with methods by Azinović *et al*. [2] and David *et al*. [31] on synthetic scene. There are two different materials on this sink (the specular top surface and the diffuse front surface). Methods by Azinović *et al*. [2] and David *et al*. [31] rely on object segmentation, resulting in artifacts in the blue circle. Our method doesn't require object segmentation as input and predicts the specular highlight in the blue circle correctly.

are of 1024×1024 resolution. All captured panorama images and outdoor environment maps are unwrapped using cubic projection to prevent gradient explosion near the two poles. The projected resolution is $6 \times 512 \times 512$.

Supplementary video. We encourage readers to view our supplementary video for a comprehensive demonstration of our inverse rendering results. We show novel view synthesis, virtual object insertion, and relighting results to demonstrate our inverse rendering quality, and add an optimization procedure as an intuitive illustration.

4.1 Evaluation of Parameter Maps Recovery

First, we evaluate the quality of the recovered light source and material maps. We show the recovered parameter maps of two real scenes in Figure 7 to demonstrate the effectiveness of MILO. Only the ceiling lights are emitting light physically (light-emitting objects) in real scenes, while other objects are reflecting light. The tables and floors are smooth (specular), while the walls, ceilings, and the curtain are rough (diffuse). These properties are correctly recovered in the parameter maps. Compared with the rendered images, specular reflections and cast shadows are correctly removed in diffuse and specular albedo maps.

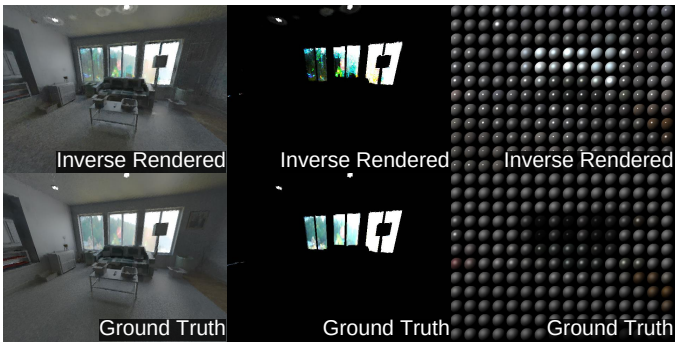
To demonstrate the advantage of recovering fully spatially varying parameters, we inverse render two synthetic scenes with ground truth object segmentation in Figure 8. In this scene, there are objects that have different materials. For example, the sink has a specular top surface (in the blue double circle) and a wooden front surface (in the green circle). We compare MILO with two state-of-the-art differentiable rendering methods by Azinović *et al*. [2] and David *et al*. [31] *, which use object segmentation as input. The re-rendered results are shown in Figure 8. Azinović *et al*. [2]'s method shares the same material parameters on each object, as a result, the re-rendered sink shows a brown top surface color learned from the front side (in the green circle), which is a wrong color. David *et al*. [31]'s method uses per-pixel diffuse albedo, but still share specular albedo and roughness parameters on each object, as a result, the top surface has wrong specular parameters and specular highlights aren't rendered correctly. Thanks to the fully spatially varying design of MILO-Net, our method doesn't rely on object segmentation as input and recovers different materials on the same object correctly.

For synthetic scenes, we evaluate parameter map recovery accuracy by comparing them with the ground truth. We visualize two example results in Figure 9. For parameter maps of light-emitting objects (k_e, k_w, and k_g), the average MSE (Mean Square Error) is 0.002. For parameter maps of materials (k_d, k_s, and k_a), we render a shading ball for each pixel (down-sampled for better visualization), and the average MSE is 0.015. The results demonstrate high accuracy of our method.

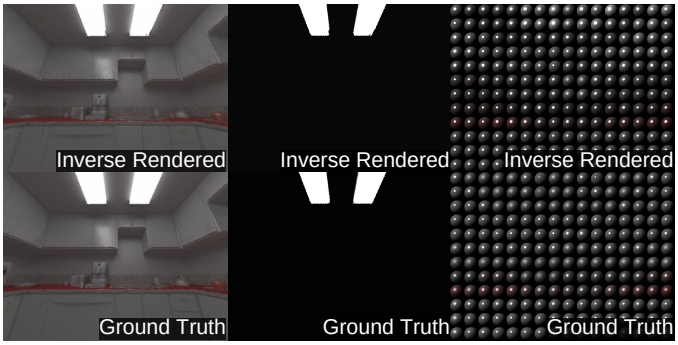
4.2 Application: Virtual Object Insertion

By editing the geometry and the predicted six parameter maps, we implement virtual object insertion. We compare our virtual object insertion quality with four state-of-the-art indoor scene inverse rendering methods, *i.e*., Li *et al*. [25], Srinivasan *et al*. [42], Azinović *et al*. [2], and David *et al*. [31]. These methods have different inputs and designs, which are listed in Table 1. We perform virtual object insertion using these five methods and calculate the error maps with respect to the ground truth as a quantitative comparison. We insert a diffuse object, a specular object, and a light-emitting object into each scene, and utilize PSNR (Peak Signal-to-noise

$*$. Both of these works are not open source. To implement their methods, we remove the three disambiguation constraints from MILO-Net, light source model from MILO-Renderer, and make some of the parameters shared within the same object.



(a) Example Scene A



(b) Example Scene B

Fig. 9. Recovery results of Light-emitting objects and materials using synthetic data. The recovered light-emitting object and surface reflectance are similar to ground truth. Please zoom in to see the details.

TABLE 3
Quantitative comparison with four methods on synthetic scenes. MILO achieves the best quantitative performance regarding all error metrics. ↑(↓) indicates larger (smaller) values are better.

Method	PSNR↑	SSIM↑	NCC↑	MSE↓
Li *et al*. [25]	24.12	0.955	0.957	0.004
Srinivasan *et al*. [42]	23.77	0.964	0.956	0.004
Azinović *et al*. [2]	14.48	0.829	0.501	0.036
David *et al*. [31]	16.54	0.827	0.775	0.022
MILO (Ours)	**26.49**	**0.964**	**0.978**	**0.002**

Ratio) [13], SSIM (Structural Similarity Index Measure) [47], NCC (Normalized Cross Correlation) [47], and MSE (Mean Square Error) as error metrics. The performances are shown in Table 3.

To intuitively demonstrate the effectiveness of our method, we visualize an example of synthetic scene in Figure 10. For real data, since ground truth for virtual object insertion is not available, we show only qualitative results in Figure 11. Methods by Li *et al*. [25] and Srinivasan *et al*. [42] preserve details on the original images well, however, high frequency reflections of the invisible area and the occlusion relationship between objects are not rendered correctly due to limited FoV (Field of View) of single/stereo input images and inserting virtual object in the image space. The inserted virtual light source object also shows no illumination on scene object due to the limitation of image based editing. From error maps of the synthetic scene, we can see that the errors concentrate on the inserted ball and surround the virtual light source. The other three methods all use path

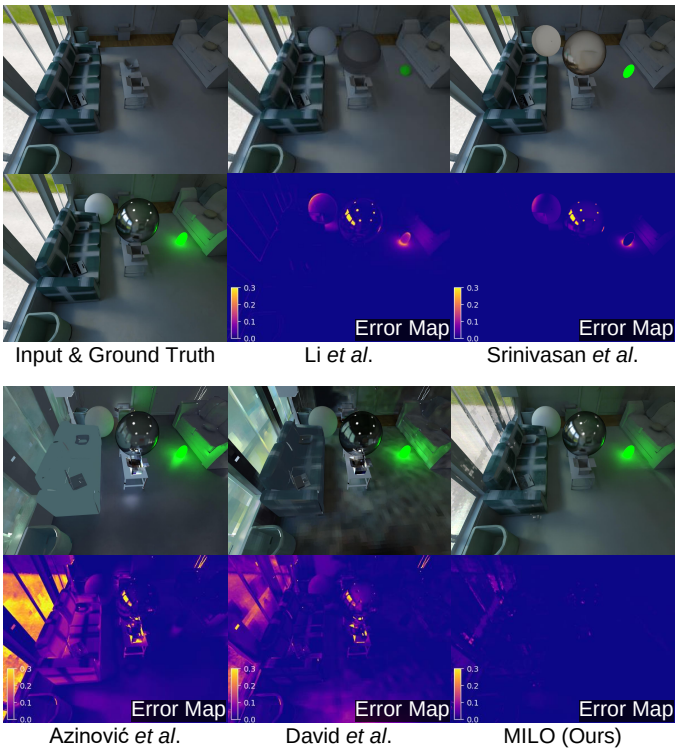


Fig. 10. Qualitative comparison for virtual object insertion on a synthetic scene with Li *et al*. [25], Srinivasan *et al*. [42], Azinović *et al*. [2], and David *et al*. [31]. Our method demonstrates lower error than other methods on error maps. Li *et al*. [25]'s method inserts balls on image plane directly, so the balls have circular shapes. Other methods insert balls in the 3D space and use perspective camera model, so the balls have ellipse shape.

tracing architecture and render virtual light source illumination well. Methods by Azinović *et al*. [2] and David *et al*. [31] rely on object segmentation. We annotate and provide Ground Truth segmentation for them. Azinović *et al*. [2]'s method recovers materials at object detail level. Textures on the sofa are missing. For scenes with window, results from Azinović *et al*. [2] and David *et al*. [31] both show uniform emission on the window. From error maps of the synthetic scene, we can see that the errors concentrate on the window, specular ball, and the highlight area on the floor. We believe it is because lighting through the window is different from their omnidirectional light source model. The missing reflection on the floor is caused by the incorrect incident light from the window. In contrast, our light source model is designed for scenes with window. As a result, there are outdoor scene through the windows, and reflection on the floor is rendered correctly. For scenes without window, David *et al*. [31]'s method and our method achieve comparable results, however, we don't require object segmentation as input. For scenes with window, only our method renders all virtual objects realistically, and produces complex cast shadows, multi-bounce effects, and virtual light source illumination correctly.

4.3 Application: Relighting

We verify the effectiveness of MILO-Net by showing a relighting application. In our method, finite material constraint and reflectance similarity constraint are implemented

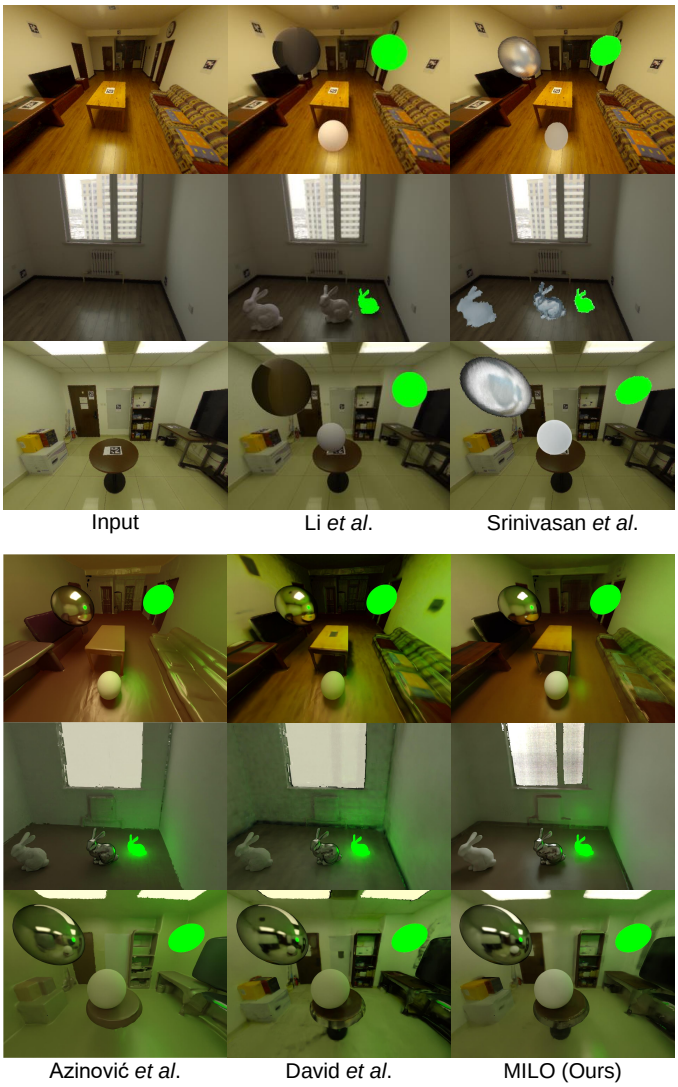


Fig. 11. Qualitative comparison for virtual object insertion on three real scenes with Li *et al*. [25], Srinivasan *et al*. [42], Azinović *et al*. [2], and David *et al*. [31]. Our method produces more photorealistic cast shadows, multi-bounce effects, and virtual light source illumination. For the scene with window, our method recovers outdoor scene through the window and renders reflection on the floor correctly.

(a) Relighting Results without MILO-Net (MLPs only)

(b) Relighting Results with MILO-Net

Fig. 12. The scene is inversely rendered using parameter maps predicted with and without MILO-Net respectively. Without MILO-Net, the scene light source cannot be fully turned off, and the walls and ceiling are emitting abnormal light. With MILO-Net, the shading on the wall and cast shadow on the floor are more realistic.

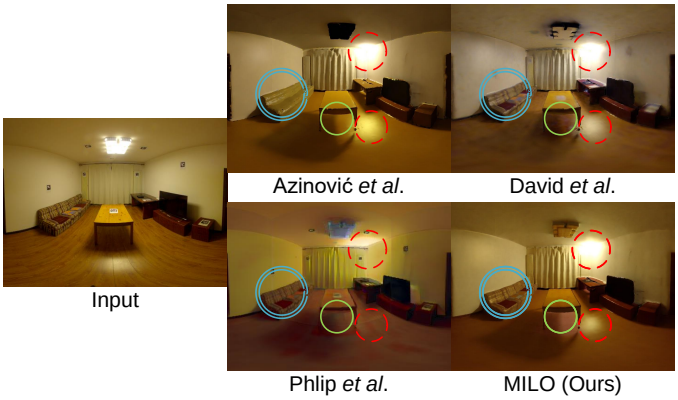


Fig. 13. Qualitative comparison with methods proposed by Azinović *et al*. [2], David *et al*. [31], and Philip *et al*. [36] for relighting on a real scene. Our method renders virtual light illumination and specular reflection (red dashed circle), cast shadow (green circle) correctly and preserves details on scene objects (blue double circle).

in MILO-Net. Without MILO-Net, we use two MLPs with position encoding to generate all six parameter maps directly. In this way, the two disambiguation constraints are disabled. We can "turn off" the scene light sources by setting its emission map to zero, and add virtual light-emitting objects to the scene, which illuminate the room with different lighting. We inversely render a real scene using MLP (in Figure 12 (a)) and MILO-Net (in Figure 12 (b)) to predict parameter maps respectively. As shown in the results, the re-rendered images with scene light source on (left) are similar to MLP. However, without MILO-Net, the scene light source is not turned off correctly (middle), the walls are too bright in relighted image (right). These artifacts are caused by diffuse-emission ambiguity. In Figure 12 (a), the walls and ceiling are emitting light. With a lower diffuse albedo correspondingly, the re-rendered results can be the same. However, when we turn off the original light source, the reflection light from the walls keeps emitting abnormally. In contrast, The walls are classified as reflecting light correctly with the help of disambiguation constraints in MILO-Net. As a result, these two images are both rendered correctly.

We compare our relighting quality with methods proposed by Azinović *et al*. [2], David *et al*. [31], and Philip *et al*. [36] †. The results are shown in Figure 13. Azinović *et al*. [2]'s method assumes uniform material on each object, which erases the detailed textures on the sofa (in blue double circle). In contrast, our method renders the texture correctly thanks to the fully spatially varying design on every material parameters. David *et al*. [31]'s method provides much detailed textures, however, the cast shadow of the original ceiling light under the table is not removed cleanly (in green circle). We think it is because this method optimizes diffuse albedo without constraints, which is not robust to deviations in light source and reflection model. In contrast, our method removes cast shadow more cleanly and is more robust to noises thanks to finite material

†. Scanned geometry and HDR images are provided to this method as input, which are better than MVS geometry and RAW images in the original settings. Each panoramic image is split into 6 perspective images.

Fig. 14. Exporting recovered scenes to other renderers. Mitsuba 3 and MILO-Renderer generate similar images in scene re-rendering.

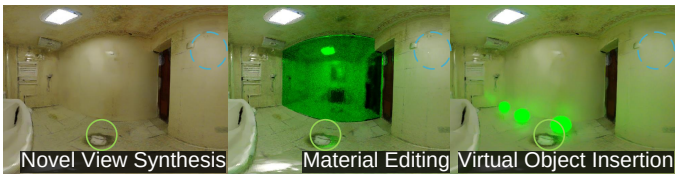
Fig. 15. Effectiveness of sufficient sampling constraint. There is a light source on the top, a white diffuse plane on the left, and a red diffuse plane on the right (left image). Color bleeding occurs without sufficient sampling constraint (middle image). The diffuse albedo is recovered correctly only with sufficient sampling constraint (right image)

constraint. Philip *et al*. [36]'s method shifts cast shadows on the floor correctly and preserves detailed textures. However, the specular reflection is much weaker, and the virtual light illumination is less contrastive (in dashed red circles). We think it is because this method uses image-based neural network to render the image, which has limitations in generalization ability and may not preserve physical properties. In contrast, our method generates image directly from path tracing based renderer, which renders virtual light illumination and specular reflection correctly.

4.4 Discussion

Exporting scene to other renderers. For compatibility with the existing engines in scene re-rendering, we export the recovered scenes by simplifying the light-emitting object model. The existing rendering engines, *e.g.*, Mitsuba [15] and Blender [7], lack support of our light source model and the importance sampling technique accordingly. They only support objects with uniform emission intensity and environment maps, so we split the ceiling lights and windows from the geometry by a threshold. For ceiling lights, we assign the average emission intensity to the objects. For windows, we remove the geometry to let the light passing through and use environment map model for outdoor scenes. As shown in Figure 14, we re-render the optimized scenes using Mitsuba 3. The result images are similar to ours. However, MILO-Renderer is the only option for scene reconstruction because our light source model is essential in the optimization procedure but it is not supported by other renderers.

Effectiveness of sufficient sampling constraint. Color bleeding is a common problem in many inverse rendering methods [31], [42], [50]. It is usually caused by two diffuse surfaces inter-reflecting light to each other, resulting in albedo maps with mixed color. We build a simplified scene in Figure 15 (left) to demonstrate this phenomenon. There is a light source on the top and two diffuse planes facing each other. Due to inter-reflection, the left plane looks pink. In our MILO-Net, the sufficient sampling constraint is used to deal with the color bleeding problem. As shown in Figure 15 (middle), the diffuse albedo of the left wall is a mixed color of white and red without sufficient sampling constraint. We believe this is because the diffuse albedo parameters k_d of two planes are functions of light source radiance, which is provided by sufficient sampling constraint. Our rendering result in Figure 15 (right) shows that, with sufficient sampling constraint, the light source radiance is recovered correctly, and the diffuse albedo parameters of the two walls are clean without color bleeding.



(a) Inverse Rendering Using a Single Input Image

(b) Inverse Rendering Using 22 Input Images

Fig. 16. The scene is inversely rendered using a single image and 22 images as input respectively. When using a single image as input, the specular reflection on the wall is missing (dashed blue circle), and there is a dark area on the floor where there was specular highlight in the input image (green circle). A larger number of input images produces higher quality on all three applications.

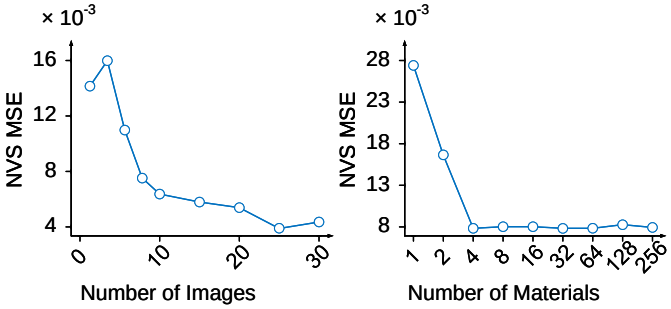


Fig. 17. Influence of image number and material number. MILO produces robust results as long as image number is not too small (≥ 10) and material number is not too small (≥ 4).

Influence of hyperparameters. There are two hyperparameters in our method: The number of input images and the number of materials in the material template.

According to sufficient sampling constraint, if we use too few images as input, multi-bounce ambiguity will cause errors in re-rendered images of edited scenes. As shown in Figure 16, results on novel view synthesis, material editing, and virtual object insertion are all better in Figure 16 (b) (using 22 input images) than Figure 16 (a) (using a single

input image). In contrast, taking too many images is time-consuming for the photographer, and limits the application. We compare novel view synthesis quality with respect to the input image number in Figure 17 (left). The re-rendered images achieve robust quality as long as the input image number is not too small (≥ 10), so we capture 20–30 images for all scenes.

In the material template, there is a hyperparameter Q denoting the total number of materials. According to finite material constraint, if the material template includes too many materials (Q too large), the effect of disambiguation will be reduced, and it will also consume more hardware resources. In contrast, too few materials (Q too small) can hardly represent real scenes. We compare novel view synthesis quality with respect to material number in Figure 17 (right). The re-rendered images achieve robust quality as long as the material number is not too small (≥ 4). When Q is larger than the number of materials in the scene, multiple redundant materials will converge to the same one, so choosing a relatively larger Q will not affect the quality. We use 64 materials to recover all scenes.

5 Conclusion

We propose **MILO**: **M**ulti-bounce **I**nverse rendering for indoor scene with **L**ight-emitting **O**bjects. Through the proposed light-emitting object model, we can edit indoor scenes with more complex light sources. By designing three disambiguation constraints, we can eliminate most errors in re-rendered images. The results on both synthetic and real data demonstrate the advantages of MILO comparing with existing methods in virtual object insertion, material editing, relighting tasks, and so on.

The limitations of MILO are as follows: First, MILO requires scanned geometry as input, which costs more than image-based methods. The scanned geometries are of lower resolution than images and contain artifacts, which limit the inverse rendering quality. Second, we train MILO-Net for each scene on the fly, which requires the neural network to remember the whole scene. As a result, some of the details are lost (*e.g.*, the texture on the floor in the middle row of Figure 11) due to the limited representation power of neural networks. It also consumes more computational power than learning-based methods. These limitations will be further investigated in our future work.

Acknowledgment

This work is supported by National Natural Science Foundation of China under Grant No. 62136001, 62088102, and Shenzhen Collaborative Innovation Program (CJGJZD2021048092601003).

References

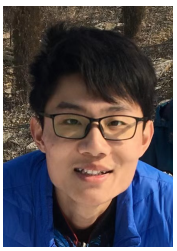
[1] Michael Ashikhmin and Peter Shirley. An anisotropic phong brdf model. *Journal of Graphics Tools*, pages 25–32, 2000.
[2] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. In *Proc. of Computer Vision and Pattern Recognition*, 2019.
[3] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1670–1687, 2014.
[4] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 218–233, 2003.
[5] Sai Bi, Xiaoguang Han, and Yizhou Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, pages 1–12, 2015.
[6] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David J. Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In *Proc. of European Conference on Computer Vision*, 2020.
[7] Blender Online Community. Blender: A 3D modelling and rendering package, 2022.
[8] Nicolas Bonneel, Balazs Kovacs, Sylvain Paris, and Kavita Bala. Intrinsic decompositions for image editing. In *Computer Graphics Forum*, 2017.
[9] Fatih Calakli and Gabriel Taubin. SSD: smooth signed distance surface reconstruction. *Comput. Graph. Forum*, 30(7):1993–2002, 2011.
[10] Chong Cao, Feng Lu, Chen Li, Stephen Lin, and Xukun Shen. Makeup removal via bidirectional tunable de-makeup network. *IEEE Transactions on Multimedia*, pages 2750–2761, 2019.
[11] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, pages 7–24, 1982.
[12] https://www.dotproduct3d.com/dot3dpro.html.
[13] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, pages 800–801, 2008.
[14] https://www.intelrealsense.com/depth-camera-d455/.
[15] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.
[16] James T Kajiya. The rendering equation. In *Proc. of ACM SIGGRAPH*, 1986.
[17] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, page 3, 2013.
[18] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David A. Forsyth. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics*, pages 32:1–32:15, 2014.
[19] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. of Computer Vision and Pattern Recognition*, pages 3907–3916. IEEE Computer Society, 2018.
[20] Young Min Kim, Sangwoo Ryu, and Ig-Jae Kim. Planar abstraction and inverse rendering of 3d indoor environment. *IEEE transactions on visualization and computer graphics*, 2019.
[21] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.
[22] Eric P. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, February 1995.
[23] Eric P Lafortune and Yves D Willems. *Using the modified phong reflectance model for physically based rendering*. Katholieke Universiteit Leuven. Departement Computerwetenschappen, 1994.
[24] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics*, pages 222:1–222:11, 2018.
[25] 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 *Proc. of Computer Vision and Pattern Recognition*, 2020.
[26] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proc. of International Conference on Computer Vision*, pages 7707–7716. IEEE, 2019.
[27] Matthew M. Loper and Michael J. Black. Opendr: An approximate differentiable renderer. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proc. of European Conference on Computer Vision*, pages 154–169. Springer, 2014.
[28] Yupeng Ma, Xiaoyi Feng, Xiaoyue Jiang, Zhaoqiang Xia, and Jinye Peng. Intrinsic image decomposition: A comprehensive review. In Yao Zhao, Xiangwei Kong, and David Taubman, editors, *International Conference on Image and Graphics Processing*, pages 626–638. Springer, 2017.
[29] https://www.meshlab.net/.
[30] 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 *Proc. of European Conference on Computer Vision*, 2020.
[31] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and lighting reconstruction for complex indoor scenes with texture-space differentiable rendering. In *Eurographics Symposium on Rendering*, 2021.
[32] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: a retargetable forward and inverse renderer. *ACM Transactions on Graphics*, pages 203:1–203:17, 2019.
[33] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.
[34] https://www.openimagedenoise.org/index.html.
[35] 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. *arXiv preprint arXiv:1912.01703*, 2019.
[36] Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Trans. Graph.*, 40(5):194:1–194:18, 2021.
[37] https://polyhaven.com/hdris.
[38] Ravi Ramamoorthi and Pat Hanrahan. On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. *Journal of the Optical Society of America A*, pages 2448–2459, 2001.
[39] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, Feb. 2018.
[40] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D Castillo, and David W Jacobs. SfSNet: Learning shape, reflectance and illuminance of faces in the wild. In *Proc. of Computer Vision and Pattern Recognition*, 2018.
[41] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.
[42] Pratul P. Srinivasan, Ben Mildenhall, Matthew Tancik, Jonathan T. Barron, Richard Tucker, and Noah Snavely. Lighthouse: Predicting lighting volumes for spatially-coherent illumination. In *Proc. of Computer Vision and Pattern Recognition*, 2020.
[43] John E Stone, David Gohara, and Guochun Shi. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, page 66, 2010.
[44] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.
[45] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proc. of Neural Information Processing Systems*, 2020.
[46] https://theta360.com/en/about/theta/z1.html.
[47] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, pages 600–612, 2004.
[48] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3d spatially-varying lighting. In *Proc. of International Conference on Computer Vision*, 2021.
[49] Yair Weiss. Deriving intrinsic images from image sequences. In *Proc. of International Conference on Computer Vision*, 2001.
[50] Yizhou Yu, Paul E. Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In Warren N. Waggenspack, editor, *Proc. of ACM SIGGRAPH*, pages 215–224. ACM, 1999.
[51] Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. Monte carlo estimators for differential light transport. *ACM Transactions on Graphics*, pages 78:1–78:16, 2021.
[52] Edward Zhang, Michael F. Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics*, pages 174:1–174:14, 2016.
[53] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *Proc. of Computer Vision and Pattern Recognition*, 2017.
[54] Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. Physics-based differentiable rendering: from theory to implementation. In *Proc. of ACM SIGGRAPH*, pages 14:1–14:30. ACM, 2020.
[55] Hao Zhou, Xiang Yu, and David Jacobs. GLoSH: Global-local spherical harmonics for intrinsic image decomposition. In *Proc. of International Conference on Computer Vision*, 2019.

Bohan Yu received the BS degree from Peking University in 2021. He is currently pursuing Ph.D. degree with Professor Boxin Shi at the Camera Intelligence Lab, Peking University. His research interest lies on computational photography and inverse rendering.

Siqi Yang received the BS degree from Peking University in 2022. He is currently pursuing Ph.D. degree at Peking University. His research interests include computational photography, inverse rendering and outdoor relighting.

Xuanning Cui received the BS degree from Peking University in 2022. He is currently pursuing ME degree at Peking University. His research interests include inverse rendering and outdoor relighting.



Siyan Dong is a Ph.D. candidate at Shandong University. His research interests mainly include 3D reconstruction, visual localization, scene understanding, and robotics.

Baoquan Chen is an Endowed Boya Professor of Peking University. His research interests generally lie in computer graphics, computer vision, visualization, and human-computer interaction. Chen received an MS in Electronic Engineering from Tsinghua University, Beijing, China, and a second MS and then PhD in Computer Science from the State University of New York at Stony Brook, New York, U.S.A. For his contribution to spatial data visualization, he was elected IEEE Fellow in 2020. He was inducted to IEEE Visualization Academy.

Boxin Shi received the BE degree from the Beijing University of Posts and Telecommunications, the ME degree from Peking University, and the PhD degree from the University of Tokyo, in 2007, 2010, and 2013. He is currently a Boya Young Fellow Assistant Professor and Research Professor at Peking University, where he leads the Camera Intelligence Lab. Before joining PKU, he did research with MIT Media Lab, Singapore University of Technology and Design, Nanyang Technological University, and National Institute of Advanced Industrial Science and Technology from 2013 to 2017. His papers were awarded as Best Paper Runner-Up at International Conference on Computational Photography 2015 and selected as Best Papers from ICCV 2015 for IJCV Special Issue. He has served as an editorial board member of IJCV and an area chair of CVPR/ICCV. He is a senior member of IEEE.