Real-Time Lighting Estimation for Augmented Reality via Differentiable Screen-Space Rendering

Celong Liu, Lingyu Wang, Zhong Li, Shuxue Quan, and Yi Xu

Abstract—Augmented Reality (AR) applications aim to provide realistic blending between the real-world and virtual objects. One of the important factors for realistic AR is the correct lighting estimation. In this article, we present a method that estimates the real-world lighting condition from a single image in real time, using information from an optional support plane provided by advanced AR frameworks (e.g., ARCore, ARKit, etc.). By analyzing the visual appearance of the real scene, our algorithm can predict the lighting condition from the input RGB photo. In the first stage, we use a deep neural network to decompose the scene into several components: lighting, normal, and Bidirectional Reflectance Distribution Function (BRDF). Then we introduce differentiable screen-space rendering, a novel approach to providing the supervisory signal for regressing lighting, normal, and BRDF jointly. We recover the most plausible real-world lighting condition using Spherical Harmonics and the main directional lighting. Through a variety of experimental results, we demonstrate that our method can provide improved results than prior works quantitatively and qualitatively, and it can enhance the real-time AR experiences.

Index Terms—Mixed/augmented reality, rendering, scene understanding, light estimation, real time

1 Introduction

Lighting estimation from an image is a challenging problem. It is also an important problem in Augmented Reality (AR) that ensures the shading and shadow consistency between a virtual object and its surrounding real environment. The forward rendering process is usually deterministic. To form an image, the intrinsic properties of the scene are required, such as geometry, material, camera information, and lighting condition. By applying the rendering function [1], an image can be generated. However, the inverse process is ill-posed. Recovering the lighting, or any other properties from a single image is severely under-constrained. To reduce ambiguity, existing solutions usually place a probe (an object with known shape and/or material) in the scene to assist the computation (e.g., a metal ball [2] or objects with known shape [3]). The requirement of inserting such probe into the scene becomes the major limitation of the practical applications.

Researchers have also explored probe-free approaches, such as using RGB-D input [4], [5] and multi-view images [6], [7], [8], [9]. Recent works [10], [11] have proposed end-to-end learning approaches that directly predict lighting from an input image. While these methods have demonstrated their capability of predicting relatively accurate lighting estimation, we argue that they are not suitable for AR applications, because real-time AR requires low-cost execution and rendering.

In this paper, we aim to estimate the distribution of lighting from all directions, represented by Spherical Harmonics (SH), in real time. Plane detection is available in advanced AR frameworks such as ARCore and ARKit. In applications developed using these frameworks, it is quite common for users to place virtual objects on a real support plane that is detected by these frameworks (e.g., table or floor). Therefore, we propose a practical approach for estimating lighting in real time from a single RGB image. We utilize detected planes as an optional geometry cue and find that it can improve lighting prediction. Our deep learning-based method takes a single image and an optional detected plane as the input, and outputs the 5th-order SH coefficients of the lighting. Our approach has two main advantages:

- We propose a low cost differentiable screen-space rendering algorithm that can render a scene with limited geometry and BRDF information under SH lighting. More importantly, this renderer can be used in the autoencoder architecture for a deep learning task.
- SH is a low-dimensional representation for the lighting (108 values for 5th-order RGB SH lighting) and can be inferred with a compact encoder-decoder architecture.

2 Related Work

In this section, we briefly review the probe-based and probe-free lighting estimation algorithms, and the differentiable rendering on scene understanding tasks.

- *Celong Liu is with Bytedance Inc., Mountain View, CA 94041 USA. E-mail: celong.liu@bytedance.com.*
- *Lingyu Wang is with Augmented Reality, OPPO US Research Center, Palo Alto, CA 94303 USA. E-mail: lingyu.wang@oppo.com.*
- *Zhong Li, Shuxue Quan, and Yi Xu are with OPPO US Research Center, InnoPeak Technology Inc., Palo Alto, CA 94303 USA. E-mail: {zhong.li, shuxue.quan, yi.xu}@oppo.com.*

Manuscript received 17 February 2021; revised 10 December 2021; accepted 7 January 2022. Date of publication 11 January 2022; date of current version 28 February 2023.
(Corresponding author: Celong Liu.)
Recommended for acceptance by S. Zollmann.
Digital Object Identifier no. 10.1109/TVCG.2022.3141943

Fig. 1. Virtual objects rendered into a real scene using estimated lighting. The image on the left is taken by a mobile phone. We estimate the lighting condition from this image and render the virtual objects into the composed scene shown in the middle image. Detailed rendering effects such as soft shadows and glossy surfaces are shown in the zoomed-in figures on the right.

2.1 Light Probe-Based Methods

Placing a physical light probe is a straightforward method for estimating an environment's illumination. A mirrored ball is used in Debevec [2] to capture the lighting at a target position by analyzing the pattern distribution on the surface of the ball. Later, researchers have extended the concept of light probe to objects with known surface geometry [12], [13], objects with known BRDF [14], objects with both known geometry and BRDF [15], arbitrarily shaped objects with homogeneous materials [16], and human faces [17], [18], [19].

In this paper, we leverage the Matterport3D dataset [20] to generate groundtruth normal and SH coefficients to train a network for lighting estimation without the need of light probes. We have also captured a large amount of outdoor data similar to DeepLight [21] to generate additional groundtruth data.

2.2 Probe-Free Light Estimation Methods

In recent years, deep learning has shown its ability in scene understanding tasks. It enables probe-free approaches for lighting estimation directly from an input image.

LeGendre *et al.* [21] propose an end-to-end neural network method that predicts lighting from a single image. They use a Google Pixel smartphone to capture pictures containing spheres of different BRDF measurements as the light probes. In order to achieve real-time performance, they compress the environment map into a small size (32x32), and train a network to predict the environment map by using an adversarial loss. However, relying on the adversarial loss makes their result less stable. Garon *et al.* [22] predict the 5th-order spatially varying SH lighting on each pixel. Their approach is a data-driven end-to-end solution without relying on geometry information. Our approach outperforms theirs around the foreground object region due to better understanding of the scene. However, our performance towards the boundary of the image is worse since our approach only predicts a global lighting. Please see Section 4.2 for details. Gardner *et al.* [23] use parametric Gaussian-like lighting to achieve spatial variation. Lighthouse [24] predicts the amount of volumetric lighting from a stereo image pair.

Song and Funkhouser [11] propose an end-to-end network that directly maps an input image to a High-Dynamic Range (HDR) environment map by using geometric warping and generative adversarial networks (GANs). They warp an input image to a partial environment map, use a GAN to fill the missing regions, and then use another GAN to generate the HDR environment map. The predicted HDR environment map represents high quality lighting distribution that provides enough contrast and is essential to rendering realistic shadows or shadings. However, obtaining such an HDR environment map by using GANs is computationally heavy and is difficult to perform in real time. For AR applications, real-time or near real-time performance is preferred.

Chalmers *et al.* [25] train a series of stacked Convolutional Neural Networks (CNNs) that predict HDR Reflection Maps (RM) from an input image. Each CNN in the stack outputs a RM corresponding to a different roughness value. This approach requires a large memory footprint and is difficult to achieve real time lighting estimation. Other works use a deep representation of appearance [26] or use CNNs to predict the reflectance map [27], [28]. While they improve the quality compared with classic approaches, the pipelines are complicated for real-time applications.

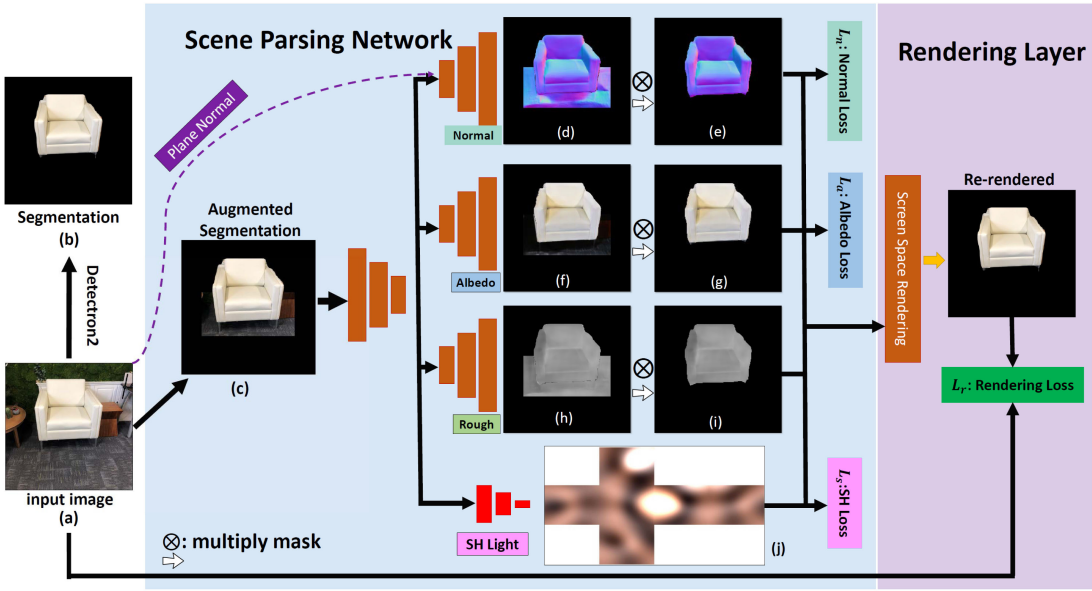


Fig. 2. Overall structure of our framework. For a single input image, we first extract the foreground, and then we augment the foreground with pixels from the support plane. *Scene Parsing Network:* the augmented foreground is fed to an initial encoder followed by a decoder to predict normal, albedos, roughness, and regress a 5th-order SH lighting. Foreground mask is applied to the predicted normal, albedos, and roughness to remove the -planar region. *Rendering Layer:* the Screen-Space Renderer will take the normal, albedos, roughness, and SH lighting as input to generate a re-rendered image of the foreground object.

Li *et al.* [29], [30] proposed a scene decomposition network that predicts normal, depth, BRDF, and lighting from a single image. They used a cascade structure to iteratively refine the predictions, but the iteration process is not easily adoptable into a real-time application.

In this paper, we use 5th-order SH lighting to approximate the environment map with minimal loss of accuracy. This lowers the prediction and rendering cost and allows real-time performance.

2.3 Differentiable Rendering

As the deep learning community grows, many tasks are bridging computer vision and computer graphics. Lighting estimation is one such example. The differentiable renderer is used in joint shape and BRDF reconstruction [31], joint illumination and BRDF estimation [32], and intrinsic image decomposition [33]. To propagate the changes in the observed image to those 3D scene parameters (shape, lighting, BRDF, camera pose, etc.), a number of existing techniques utilize differentiable rendering [34], [35], [36], [37]. However, they are designed on mesh-based rendering systems (e.g., rasterization or ray-tracing) and it is difficult to acquire a high-quality mesh for general scenes. They also require all the 3D geometric properties (coordinates, normal, and BRDF) to perform accurate rendering. In such systems, the reconstruction tasks are always intertwined so that all properties must be reconstructed together. In order to reconstruct a single property, we must also reconstruct all other properties as a consequence. However, simultaneously reconstructing all properties has a number of drawbacks. In terms of supervised learning, 3D labeling is too expensive and often not accurate. In terms of unsupervised learning, the amount of available data is not enough to learn the intrinsic manifold between properties. A recent survey of neural rendering by Tawari *et al.* [38] mentions that using the deferred shading model in neural networks has the advantage on reducing errors. Screen-space shading, as the most common implementation for the deferred shading, can turn a classical mesh-based rendering to computations on per-pixel attributes such as normal and reflectance. Nalbach *et al.* [39] use a CNN to synthesize appearance from per-pixel attributes and show improvements on speed and quality. Hence, in this paper, we develop a light-weight differentiable screen-space renderer. Although it still needs several properties (lighting, normal, albedos, and roughness) to perform rendering, it requires fewer properties than other methods. All these properties are defined per-pixel and the cost of rendering is very low, providing a more computationally efficient solution.

3 Our Method

A single RGB image is the input to our method. Building upon the recent success of differentiable rendering-based autoencoders, we tackle the problem by training an autoencoder with multiple decoder blocks whose outputs are used for screen-space rendering. In this section, the components of this autoencoder are described. The overall structure of our framework is shown in Fig. 2.

3.1 Dataset Construction

To support the training of our deep autoencoder, a large collection of images and their corresponding groundtruth lighting are needed. Obtaining groundtruth lighting for an image can be expensive. One approach is to use a physically-based rendering engine to synthesize images and lighting simultaneously. However, the networks trained with synthetic data usually do not easily generalize well to real data.

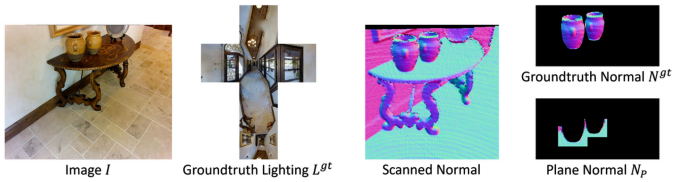


Fig. 3. Indoor groundtruth data generation. We generate groundtruth lighting and normal for images that contain a support plane in Matterport3D [20]. We fit 5th-order SH basis to the panoramic environment map to generate the groundtruth SH lighting.

Fig. 4. The outdoor data capturing rig and a sample data. (a): the capturing rig; (b) an example photo taken by the rig; (c) processed data (the top is the input image to the network and the bottom is the groundtruth 5th-order SH lighting).

Instead, by leveraging the Matterport3D dataset [20], we generate such training data from the HDR RDB-D images. In the Matterport3D dataset, there are 194,400 RGB-D images calibrated with 10,800 panoramas for a variety of indoor scenes. In fact, the panoramas are the HDR environment maps and can be regarded as the groundtruth lighting. We apply a spherical convolution [40] to extract the 5th-order SH coefficients of these environment maps as the groundtruth lighting L^{gt}.

We use the same preprocessing as Song and Funkhouser [11] to select images from Matterport3D. A selected image should contain a horizontal support surface ($n_z > \cos(\pi/8)$). In the meanwhile, this support surface should have a semantic label $\in$ {floor, furniture} and there should be one or more objects on top of the surface. Then for each image I, we transform the objects to the screen-space (camera coordinate) and calculate the screen space normal for the objects and support plane. Additionally, the SH coefficients are transformed to the screen space accordingly. To this end, the necessary groundtruth data are constructed for an image I: the groundtruth SH coefficients L^{gt}, the groundtruth normal of object N^{gt}, and the normal of planar region N_P. Overall, we have generated 109,042 $\{I, L^{\text{gt}}, N^{\text{gt}}, N_P\}$ samples and Fig. 3 shows an example.

Note that in the training stage, the plane's normal N_P is extracted from the groundtruth data directly. While in testing, N_P will be estimated from the input image by the AR framework (in this paper, we use ARCore). All N_Ps discussed here are in the screen-space coordinates.

Since Matterport3D only contains indoor data, we have built a capturing rig similar to the one in DeepLight [21] to collect data for outdoor scenes. As shown in Fig. 4, for each frame captured by this setup, we extract the three spheres [21] and reconstruct the groundtruth environment map. We have developed a mobile application that allows controlling the exposure of the camera. According to Debevec [2], an HDR environment map can be obtained by capturing images on a mirrored ball with three different exposures. For each scene, we hold the rig still and use the mobile application to capture three images. The other two balls are used to refine the obtained HDR environment map following the HDR map correction approach in Murmann *et al.* [41]. Then the groundtruth 5th-order SH coefficients are generated for the image similar to the indoor dataset creation. Overall, we have collected 89,458 $\{I, L^{\text{gt}}\}$ samples and Fig. 4c show an example. Note that this outdoor dataset does not contain groundtruth normal information. We use it to fine-tune the model after training the model with the indoor data. More details about the training process can be found in Section 3.7.

3.2 Network Architecture

We describe the architecture of our autoencoder for regressing lighting from an input image I in this section. It consists of three modules sequentially: (1) a foreground segmentation module, (2) a scene parsing autoencoder network that consists of a single initial encoder, three decoders, and regresses a 5th-order SH lighting, and (3) a rendering layer incorporating a differentiable screen-space renderer to render the scene with the output from the scene parsing network. The scene parsing network contains three decoders for albedos (A), roughness (R), and surface normal (N). These decoders along with the SH lighting regressor all share the features extracted from the same encoder. The intuition behind this is that since the normal, BRDF parameters, and lighting are all intrinsic scene properties, by using a common encoder for feature sharing can reduce the number of parameters in the network and alleviate over-fitting.

3.3 Augmented Foreground Segmentation

Background content is usually small and even distorted by the camera projection towards the image boundary. Hence, we use foreground objects around the image center to predict lighting parameters. To segment the foreground, we use Detectron2 [42] to detect common objects. From the COCO dataset, we select 65 labels (e.g., banana, vase, tree, house, etc.), which are most likely to be found on a table or the ground. After detection, we only select those objects that have 95% of the segmented pixels located in the central 70% of the image. Fig. 5b gives such an example.

Before the scene parsing network, we extract screen space normal for the support plane using camera pose and plane output and directly connect it to the normal decoder. We denote these normal as N_P. Meanwhile, we extend the objects' mask in two steps: (1) localize the lower half of the bounding box B_l, and (2) extend B_l by 1.3 in both x and y directions (Fig. 5c). The intuition behind this amplifier factor is that we want to include some planar regions under the object in the normal estimation. Then during testing, the estimated normals are more accurate under the guidance of known plane normal given by the AR framework if available. It is better to use a large factor since more pixels can be supervised. However, a larger region of interest (ROI) will likely include other objects. This will harm the design of our algorithm. We found that the value 1.3 achieves the best

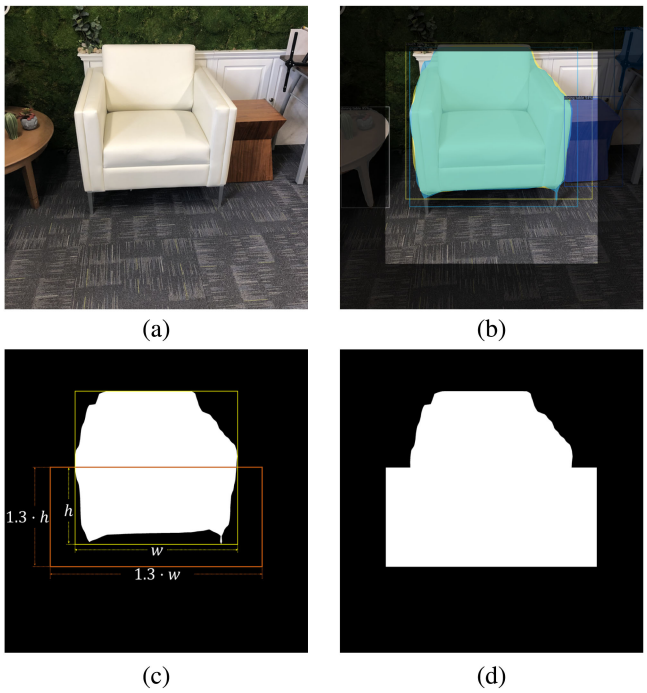


Fig. 5. Augmented foreground segmentation. (a) Input image, (b) the detection result from Detectron2, (c) the initial segmentation, and (d) the augmented segmentation.

performance. A final augmented segmentation is shown in Fig. 5d. By incorporating the passed-in accurate plane normal and the mask of the planar region, the overall quality of the normal estimation is improved.

Note that the planar region extraction is only used in the training stage using our indoor dataset. For fine-tuning on the outdoor dataset, we do not process the input image with Augmented Foreground Segmentation. In the testing phase, plane detection and foreground segmentation are both optional. When plane information is not used, zero vector will be used as the plane normal.

3.4 Scene Parsing Network

The input of the scene parsing network is the augmented segmentation I_A. We denote the mask of the object (Fig. 5c) as M and the augmented mask (Fig. 5d) as M_A. Then the mask of the planar area is represented as $M_P = M_A - M$.

Let **SceneParser(·)** be the network that contains the encoder and decoder blocks (light blue part in the middle of Fig. 2). Then the predicted normal $\tilde{N}$, albedos $\tilde{A}$, roughness $\tilde{R}$, and SH lighting $\tilde{L}$ (note that $\sim$ is used to distinguish between the estimated and true parameters) are given by:

$$\tilde{N}, \tilde{A}, \tilde{R}, \tilde{L} = \mathbf{SceneParser}(I_A, M, M_P, N_P). \quad (1)$$

The direct outputs of normal, albedos, and roughness from the decoders include the planar region of the support plane. We apply mask M on them to compute $\tilde{N}$, $\tilde{A}$, and $\tilde{R}$. In Fig. 2, they are (e), (g), and (i), respectively. Within **SceneParser(·)**, the lighting prediction module **SHEst(·)**is connected to the initial encoder named **InitEncoder(·)**. In **SHEst(·)**, two fully connected layers are used after CNN layers to regress the SH coefficients. $\tilde{L}$ can be obtained via

$$\tilde{L} = \mathbf{SHEst}(\mathbf{InitEncoder}(I_A)). \quad (2)$$

In addition, the three sub-autoencoders for normal, albedos, and roughness share the same encoder **InitEncoder(·)** as well and have their own decoders named **NormalDecoder(·)**, **AlbedoDecoder(·)**, and **RoughDecoder(·)**. Their architectures follow the U-Net [43], which reflects the physical structure of the problems. $\tilde{N}$, $\tilde{A}$, and $\tilde{R}$ can be obtained via

$$\begin{cases} \tilde{N} = \mathbf{NormalDecoder}(\mathbf{InitEncoder}(I_A), N_P) \otimes M \\ \tilde{A} = \mathbf{AlbedoDecoder}(\mathbf{InitEncoder}(I_A)) \otimes M \\ \tilde{R} = \mathbf{RoughDecoder}(\mathbf{InitEncoder}(I_A)) \otimes M \end{cases}, \quad (3)$$

where $\otimes$ is the inner product of two images. The purpose of the masking is to remove the planar region from the decoder outputs.

To predict global environmental lighting, it is vital to have a relatively large receptive field. Since **InitEncoder(·)** has 6 CNN stride-2 layers, each value in the output can be affected by the entire image. In **NormalDecoder(·)**, **AlbedoDecoder(·)**, and **RoughDecoder(·)**, the transposed CNNs are used for decoding and skip links are applied to keep details. Note that **NormalDecoder(·)** has an extra input N_P. A detailed structure of our encoder-decoder pairs is shown in Fig. 6 (left) and the detailed structure of **SHEst(·)** is shown in Fig. 6 (right).

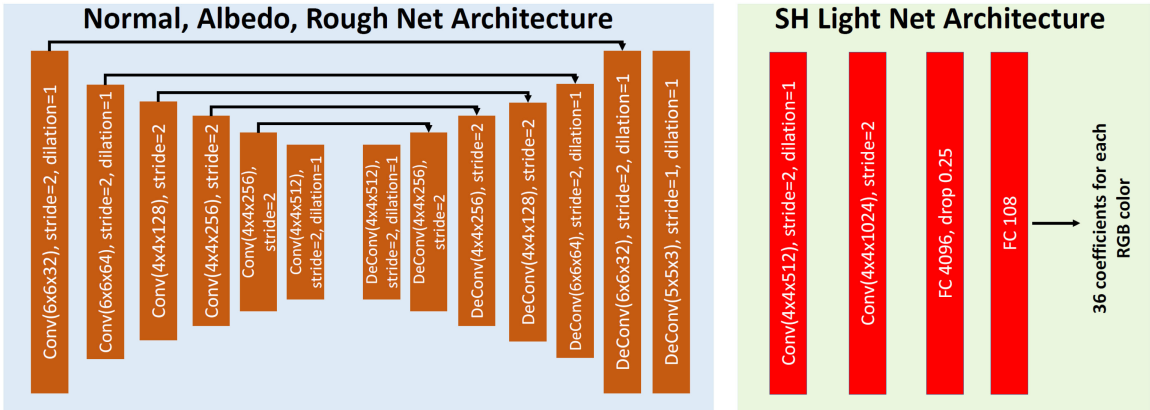


Fig. 6. The network architecture details of the encoder and decoders (left) and the lighting prediction module (right) in the SceneParser.

TABLE 1
Comparison of Errors to Groundtruth for SH Approximation Orders

	Lighting ($\log L_2$)	Image (L_2)
5th-order SH	1.56	2.49×10^{-4}
3rd-order SH	4.43	5.82×10^{-4}

Environment map 5th-order SH 3rd-order SH

Fig. 7. Comparison of images rendered with lighting approximations. Using 5th-order SH lighting can recover high-frequency lighting effects much better than using 3rd-order SH lighting.

3.5 Screen-Space Rendering

To render the object with the output from **SceneParser(·)**, we have developed an explicit solution to the rendering equation [1] that can be differentiated to support back propagation. Since the transmission effects,[1] self-luminous objects, and global illumination effects[2] are not targeted by our approach, we can simplify the rendering equation to a reflection equation:

$$L_o(\vec{X}, \vec{\omega_o}) = \int_{\Omega} f_r(\vec{X}, \vec{\omega_i}, \vec{\omega_o}) L_i(\vec{X}, \vec{\omega_i})(\vec{\omega_i} \cdot \vec{n}) d\vec{\omega_i}, \tag{4}$$

where L_o is the total spectral radiance from a particular position $\vec{X}$ on the object directed towards eye along direction $\vec{\omega_o}$, L_i is the incident radiance at $\vec{X}$ along direction $\vec{\omega_i}$, and f_r is the BRDF at $\vec{X}$. The integral is over the normal-oriented hemisphere Ω towards the normal $\vec{n}$ at $\vec{X}$.

3.5.1 Lighting

Lighting is parameterized as a 5th-order SH lighting. Since it represents a global lighting condition, the radiance is only dependent on the direction. More specifically, we can represent L_i with

$$L_i(\vec{X}, \vec{\omega_i}) = L_i(\vec{\omega_i}) = L_i(\theta_i, \phi_i) = \sum_{l=0}^{5} \sum_{m=-l}^{l} \tilde{L}_{lm} Y_{lm}(\theta_i, \phi_i), \tag{5}$$

where (θ_i, ϕ_i) are the altitude and azimuth respectively in camera coordinates, $\tilde{L}_{lm}$ is predicted SH coefficients in $\tilde{L}$, and Y_{lm} is SH basis.

3.5.2 BRDF

We use the microfacet BRDF model [44]. Following the notations above, and $\vec{h} = (\vec{\omega_i} + \vec{\omega_o})/2$, the model is defined as

$$\begin{aligned} f_r(\vec{X}, \vec{\omega_i}, \vec{\omega_o}) &= f_r(\vec{\omega_i}, \vec{\omega_o}, \tilde{N}, \tilde{A}, \tilde{R}) \\ &= \frac{\tilde{A}}{\pi} + \frac{D(\vec{h}, \tilde{R}) F(\vec{\omega_o}, \vec{h}) G(\vec{\omega_i}, \vec{\omega_o}, \vec{h}, \tilde{R})}{4(\vec{n} \cdot \vec{\omega_i})(\vec{n} \cdot \vec{\omega_o})} = \frac{\tilde{A}}{\pi} + \frac{\mathscr{B}(\theta_i', \theta_o', \tilde{R})}{4 \cos\theta_i' \cos\theta_o'}, \end{aligned} \tag{6}$$

where D, F, and G are the normal distribution, Fresnel term, and geometric term respectively. (θ_i, ϕ_i) and (θ_o, ϕ_o) are the altitude and azimuth of incident radiance direction and eye direction in local coordinates, respectively. From Eqn. (6), we can see the BRDF is *radially symmetric* or it is solely dependent on θ_i' when θ_o' is fixed. This property can be used to simplify the integral in Eqn. (4).

1. refraction, sub-surface scattering, and volumetric phenomena.
2. shadow, interreflection, and caustics.

3.5.3 Integral

Plug Eqns. (5) and (6) into Eqn. (4), and an analytical solution of the reflection equation can be derived explicitly. Since the camera's field of view (FOV) is available, $\vec{\omega_o}$ and $\cos\theta_o'$ can be determined by the pixel's position in the image, hence $L_o(\vec{X}, \vec{\omega_o})$ can be written as $L_o(\vec{p})$ where $\vec{p}$ is the pixel's position. We denote the rendered image as $\tilde{I}_R$.

Here we provide the solution's formula as

$$L_o(\vec{p}) = \sum_{l=0}^{5} \sum_{m=-l}^{l} \tilde{L}_{lm} D_{m0}^{l} \left(2\tilde{A}\Lambda_l + \frac{\pi}{2\cos\theta_o'} \tilde{\Theta}_l(\tilde{R}, \cos\theta_o') \right), \tag{7}$$

where D_{m0}^{l}, Λ_l are constant numbers and $\tilde{\Theta}_l(\tilde{R}, \cos\theta_o')$ is a 5 degree polynomial in terms of $\tilde{R}$ and $\cos\theta_o'$. The details of the derivation can be found in the Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2022.3141943. To this end, the rendering process uses only the screen-space attributes $\tilde{N}$, $\tilde{A}$, $\tilde{R}$, and $\tilde{L}$ and can be formulated as a low cost linear combination of polynomials of these attributes and it is differentiable. The derivatives of $L_o(\vec{p})$ on $\tilde{N}$, $\tilde{A}$, $\tilde{R}$, and $\tilde{L}$ can be found in the Appendix B, available in the online supplemental material.

3.5.4 Order of Spherical Harmonic Lighting

To investigate how the order of SH lighting affects the rendering quality, we compare images rendered with an environment map, a 5th-order SH lighting approximation, and a 3rd-order SH approximation (Fig. 7). Quantitative comparisons of lighting approximation and rendering errors are shown in Table 1. The two columns in Table 1 are the average error when using a representation to approximate per pixel lighting in Fig. 7 and the Mean Square Error (MSE) of the rendered images. It is evident that using 5th-order SH lighting can achieve better rendering results than using 3rd-order SH, especially around specular regions. Tunwattanapong *et al.* [45] use the 5th-order SH to recover reflectance and shape. They have observed that higher order SH coefficients (>3) respond only to the specular effects, and this is in line with our observations.

3.6 Loss Function

In this section, we describe the losses used in the entire pipeline. Training our network by minimizing the difference between the rendered image $\tilde{I}_R$ and I only will produce blurred lighting result. This is because the lighting estimation from one image is fundamentally an ill-posed problem [40]. We do not have a huge amount of data for modeling the

intrinsic mapping from image to lighting. Hence, we must utilize labeled training data to supervise intermediate components to make the lighting more determinable.

Since we can extract the groundtruth SH lighting coefficients from the environment map (see Section 3.1), the estimated SH coefficients can be supervised. From our dataset, we can obtain the normal of objects $N^{\mathtt{gt}}$ from groundtruth geometry. Thus, the predicted normal $\tilde{N}$ can be supervised by $N^{\mathtt{gt}}$. Similarly, the normal of the planar region N_P can be obtained from the plane's groundtruth geometry directly. Therefore, the predicted normal in the planar area $(\tilde{N}_A - \tilde{N})$ can be supervised by N_P. For albedo prediction, there are no groundtruth labels; therefore, we do not use supervision on roughness $\tilde{R}$. However, we can still judge the quality of an albedo map based on the following heuristics: the pixel with lower image-space gradient value is more likely to have the same albedo value as its neighboring pixels.

3.6.1 Rendering Loss

The rendering loss $\mathscr{L}_r$ measures the pixel-wise l_1 difference between input image I and rendered image $\tilde{I}_R$ within the masked region

$$\mathscr{L}_r = \|(I - \tilde{I}_R) \otimes M\|_1. \tag{8}$$

We found that using l_1-norm as a loss function helps with robustness to outliers, such as self-shadowing or extreme specular reflection in input I that are ignored in $\tilde{I}_R$.

3.6.2 SH Loss

The SH loss $\mathscr{L}_s$ is an MSE loss on the SH coefficients, it is defined as

$$\mathscr{L}_s = \frac{1}{3 \times 36} \sum_{c=1}^{3} \sum_{l=0}^{5} \sum_{m=-l}^{l} (L_{lm,c}^{\mathtt{gt}} - \tilde{L}_{lm,c})^2, \tag{9}$$

where $L_{lm,c}$, $\tilde{L}_{lm,c}$ are the predicted SH coefficients and the groundtruth SH coefficients for the cth color channel (in RGB), respectively.

3.6.3 Normal Loss

The normal loss $\mathscr{L}_n$ measures the pixel-wise l_2 difference between groundtruth normal $N^{\mathtt{gt}}$ and predicted normal $\tilde{N}$ within the masked region

$$\mathscr{L}_n = \|(N^{\mathtt{gt}} - \tilde{N}) \otimes M\|_2. \tag{10}$$

3.6.4 Planar Normal Loss

It is not feasible for planar normal loss $\mathscr{L}_{n_p}$ to enforce all pixels in the extended region $(M_A - M)$ to have the same normal as N_P, because the augmented segmentation may still contain other objects not on the plane. We require that most but not all of the pixels in $(M_A - M)$ are close to N_P. The definition of $\mathscr{L}_{n_p}$ is

$$\mathscr{L}_{n_p} = \sum_{\mathscr{Q}(\eta)} \Big\{ \|N_P - \tilde{N}^i\|^2, i \in (M_A - M) \Big\},$$

where $\mathscr{Q}(\eta)$ denotes the subset of $\{\cdot\}$ which is the top $\eta\%$ smallest elements in it. In our case, we select the top $\eta\%$ pixels that are closest to N_P and minimize their l_2 distance to N_P. In our experiments, η is set to 80 empirically.

3.6.5 Albedo Loss

The albedo loss $\mathscr{L}_a$ is defined based on the assumption of color similarity and intensity similarity among neighboring pixels. This term is inspired by the multi-scale shading smoothness property introduced by Li and Snavely [46]. It is defined as a weighted sum of l_2 terms over surrounding pixels. The weights are negative gradient magnitudes

$$\mathscr{L}_a = \sum_{i \in \tilde{A} \otimes M} (-|\nabla I|^i) \sum_{j \in \mathtt{nb}(i)} (\log \tilde{A}^i - \log \tilde{A}^j)^2,$$

where $\mathtt{nb}(i)$ denotes the 8-connected neighborhood around pixel i and ∇I is the gradient of image I.

To reduce over-fitting during training, we add an extra regularizer term to further constrain the optimization using statistical regularization on the estimated SH coefficients.

To this end, by combining the above six terms, we can define our final loss function as

$$\begin{aligned} \mathscr{L} = & \underbrace{\lambda_r \mathscr{L}_r}_{\mathtt{self-supervised}} + \underbrace{\lambda_a \mathscr{L}_a}_{\mathtt{unsupervised}} + \underbrace{\lambda_s \mathscr{L}_s + \lambda_n \mathscr{L}_n + \lambda_{n_p} \mathscr{L}_{n_p}}_{\mathtt{supervised}} \\ & + \underbrace{\lambda_{\mathtt{reg}} \sum_{c=1}^{3} \sum_{l=0}^{5} \sum_{m=-l}^{l} \tilde{L}_{lm,c}^2}_{\mathtt{regularizer}}. \end{aligned} \tag{11}$$

The roughness is not supervised directly since we cannot make an assumption similar to the albedo and we do not have groundtruth roughness either. However, the rendering loss already considers the roughness implicitly since the roughness is involved in the rendering pass. With these losses, our model is encouraged to reproduce accurate SH lighting, normal, and albedo.

The weighting coefficients in Eqn. (11) are $\lambda_r = 1.92$, $\lambda_a = 0.074$, $\lambda_s = 2.14$, $\lambda_n = 1.01$, $\lambda_{n_p} = 1.82$, and $\lambda_{\mathtt{reg}} = 2.9 \times 10^{-5}$. These parameters are fine-tuned by mixing the manual tuning and grid search tuning. This set of parameters seems to work the best so far in our experiments.

3.7 Training Strategies

The **SceneParser(·)** is first trained on the indoor dataset. We use the Adam optimizer with 2×10^{-4} as the learning rate for the **InitEncoder(·)** and 3×10^{-4} as the learning rate for the decoders. The learning rates are decreased by half after every two epochs. For this stage, the training takes 15 epochs. Then for the second stage using the outdoor dataset, the **NormalDecoder(·)**, **AlbedoDecoder(·)**, **RoughDecoder (·)** are fixed. The Augmented Segmentation will be skipped for this stage. The input image will be passed into the **InitEncoder(·)** and a zero vector will be used as the plane normal. For both the **InitEncoder(·)** and **SHEst(·)**, the initial learning rates are 2×10^{-5} and are reduced by half every two epochs. In this stage, the training takes 8 epochs.

4 Experimental Results

We first perform several ablation studies. Then we evaluate the lighting estimation results by comparing with several previous methods qualitatively and quantitatively. We also analyze run-time performance on mobile phones. Finally,

TABLE 2
Comparison of MSE to Groundtruth for Lighting and Normal Estimation

	`without plane`	`plane guide`
5th-order SH coef (10^{-2})	9.833	3.746
Normal (10^{-2})	6.591	4.184

we demonstrate other applications enabled by our framework, such as relighting and rendering interreflection.

4.1 Ablation Studies

In this section, we evaluate the models trained without the support plane, Normal Loss, or Albedo Loss via several ablation studies.

Support Plane. We first justify the necessity of using the support plane's normal for lighting and normal estimation. We train *SceneParser* on the training set of Matterport3D with and without the guidance of a support plane, we show the qualitative comparison in Fig. 8. Column `without plane` (trained without N_P) in Table 2 reports the recovering errors (MSE), which are clearly larger than those in column `plane guide` (trained with support plane N_P) for the same testing data. Thus, the guidance of N_P has a significant impact on lighting and normal estimation.

We also demonstrate how N_P affects the lighting estimation on images captured by an Android smartphone. Since we do not have the groundtruth normal or lighting condition, we show the qualitative comparison in Fig. 10. In the first example, the normals from `without plane` are incorrect. The light green part means the normal in that region are very different from other regions. In the second example, the normals from `without plane` are overly smooth. In both examples, the lighting is too dark for `without plane` results.

Normal Loss and Albedo Loss. In Table 3, we demonstrate the effectiveness of the normal loss $\mathscr{L}_n$ and the albedo loss $\mathscr{L}_a$. Without using $\mathscr{L}_n$, the accuracy of SH and normal estimation drops significantly. Without using $\mathscr{L}_a$, the performance degradation is less severe than without using $\mathscr{L}_n$. $\mathscr{L}_n$ has a larger impact on the final estimation of lighting. We speculate that it is because $\mathscr{L}_n$ is supervised while $\mathscr{L}_a$ is unsupervised. In Fig. 9, we qualitatively demonstrate how the two losses affect the predictions for a sample scene. As can be seen, the texture on the objects may mislead the estimation of normal, albedo, and roughness; leading to incorrect lighting estimation.

4.2 Evaluation Against State-of-the-Art

Evaluation of lighting estimation methods can be complicated. An intuitive way to evaluate the predicted lighting is to use the estimated lighting to render an inserted object and observe how well this virtual object is immersed with its surrounding scene. In all experiments, we use the predicted SH lighting and the main directional light to render virtual objects. The main directional light is extracted from the SH coefficients using the algorithm described by Sloan [47]. The reason behind using a directional light is that the specular effects and shadows are more obvious with it. In this paper, we use both offline and online rendering to evaluate the quality of virtual object insertion. For offline evaluations, we care more about the quantitative accuracy of shading on virtual objects, so we use the work of Mitsuba [48]. For online real-time rendering, the qualitative correctness is evaluated. To support such evaluation, we developed a real-time rendering engine using the predicted lighting. The offline rendering will be discussed in this section and the online rendering will be covered in the Mobile AR Application section (Section 4.3).

To render virtual objects into a real-world scene, we select a support planar region for virtual object placement. For images in the testing set, the support plane can be extracted from the groundtruth geometry. For images captured on a mobile device during an AR session (ARCore or ARKit), support plane information is available as well. The diffuse albedo and roughness value of the plane can be obtained from our model.

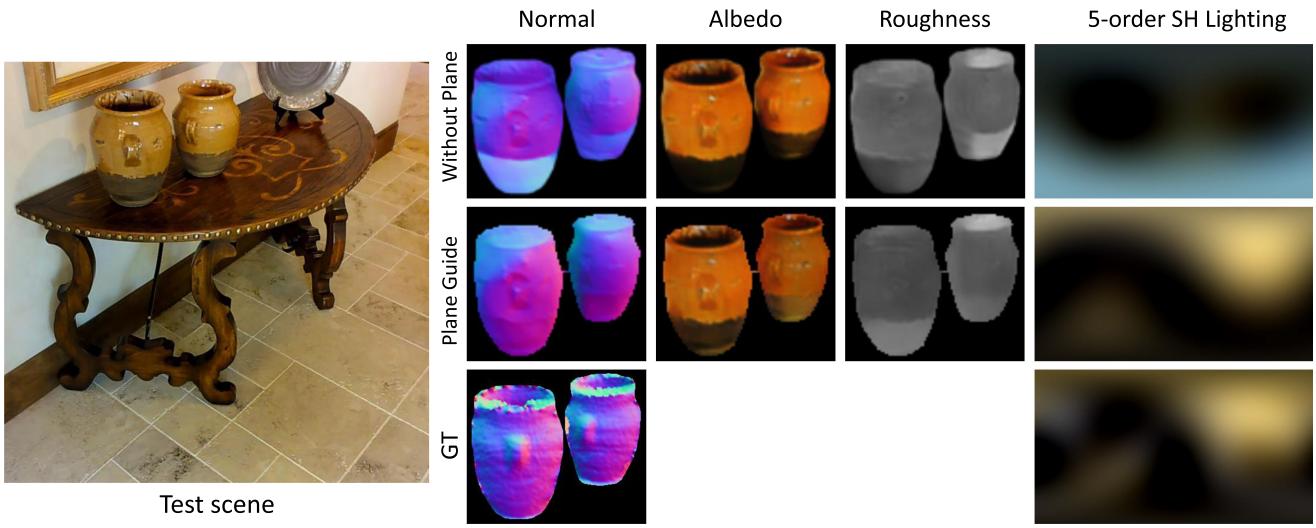


Fig. 8. Comparison of normal, albedo, roughness and lighting predictions of two networks, trained without N_P (top) and with N_P (middle), when evaluated on an input scene. Without the guidance of N_P, the texture on the objects may mislead the normal, albedo and roughness estimation and produce incorrect lighting output.

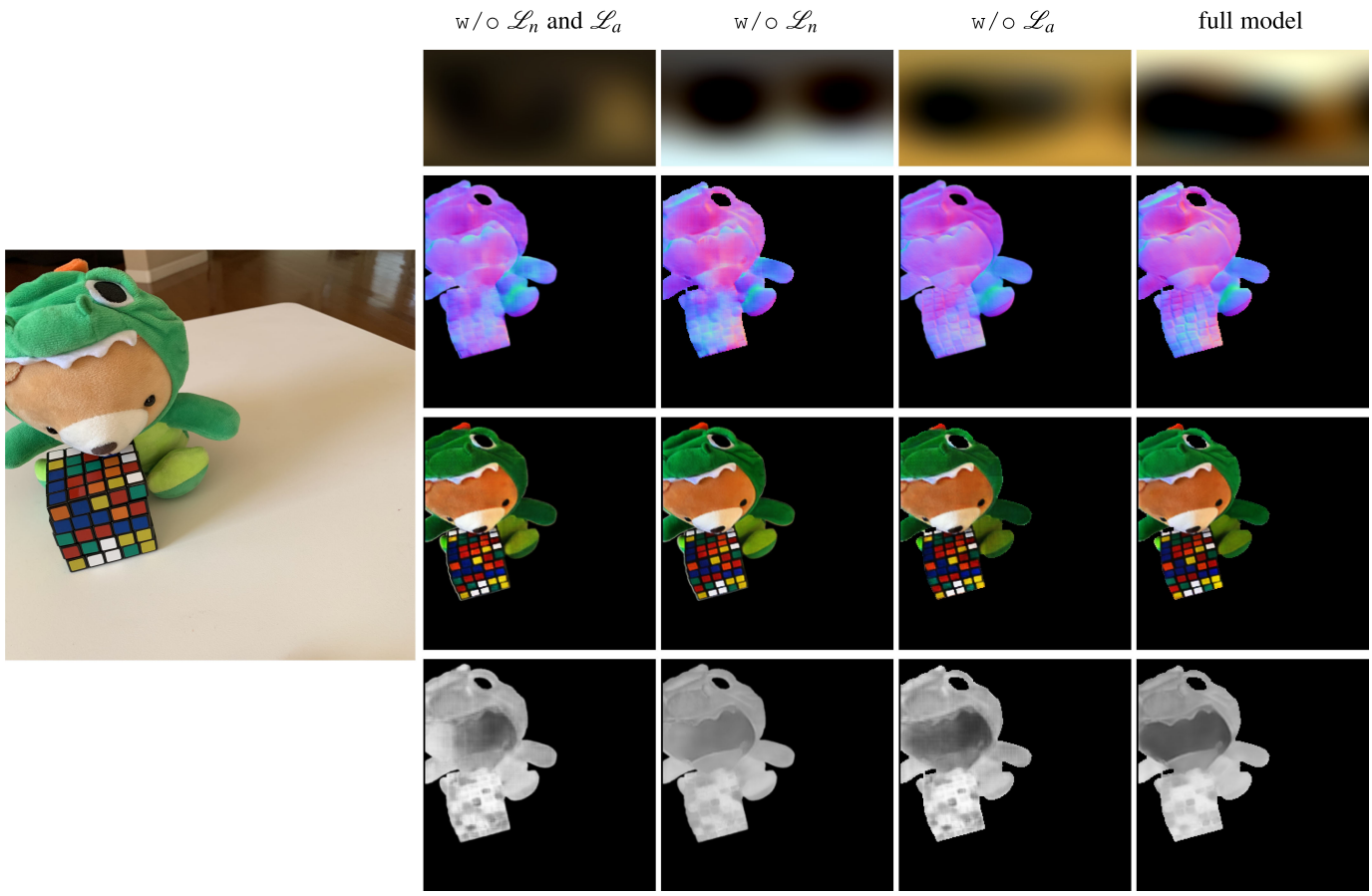


Fig. 9. Comparison of lighting estimation (1st row), normal estimation (2nd row), albedo estimation (3rd row), and roughness estimation (4th row) using four networks: trained without both $\mathscr{L}_n$ and $\mathscr{L}_a$, trained without $\mathscr{L}_n$, trained without $\mathscr{L}_a$, and the full model. Without using $\mathscr{L}_n$, the accuracy of predictions drops significantly. While without using $\mathscr{L}_a$, the performance degradation is less than without using $\mathscr{L}_n$.

We include both the support plane and the new object in our rendering process to ensure interreflection between them are properly calculated. We render two images I_{all} and I_{pl} and two binary masks M_{obj} and M_{all}. I_{all} is the rendered image of the plane and the object together. I_{pl} is the rendered image of the plane only. M_{all} is the mask covering both the cropped plane and the object. M_{obj} is the mask of the inserted object only. Let I be the original input image and I_{new} be the new image with the inserted virtual object. For the object region, we directly use the rendering result in I_{all}

$$I_{\text{new}} \odot M_{\text{obj}} = I_{\text{all}} \odot M_{\text{obj}}. \tag{12}$$

For the remaining region on the support plane, we scale the original image with the ratio of I_{all} to I_{pl}

$$I_{\text{new}} \odot (M_{\text{all}} - M_{\text{obj}}) = I \odot \frac{I_{\text{all}}}{I_{\text{pl}}} \odot (M_{\text{all}} - M_{\text{obj}}). \tag{13}$$

All operations are pixel-wise. This process utilizes the idea of ratio (or quotient images) that has been used for relighting [30], [49]. It ensures that global effects due to object-plane interaction, such as soft shadows, are visualized, while keeping intensities consistent with the overall image. It suppresses high-frequency artifacts in I_{all} caused

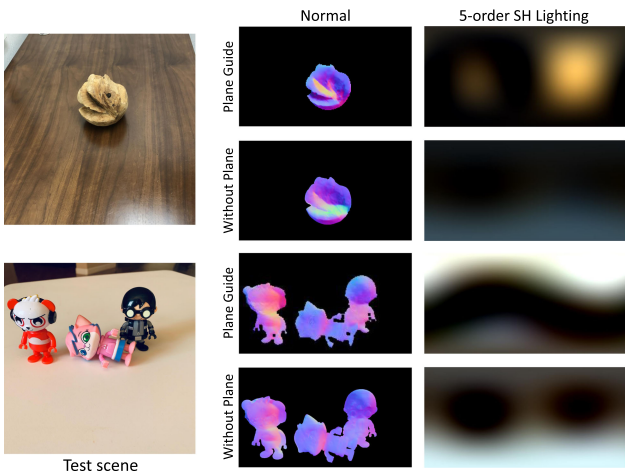


Fig. 10. Comparison of lighting and normal predictions of two networks, trained with N_P (`plane guide` rows) and without N_P (`without plane` rows).

TABLE 3
Comparison of MSE to Groundtruth for SH Prediction and Normal Prediction

	5th-order SH coef (10^{-2})	Normal (10^{-2})
without $\mathscr{L}_n$ and $\mathscr{L}_a$	5.527	4.952
without $\mathscr{L}_n$	5.284	4.879
without $\mathscr{L}_a$	3.932	4.271
full model	3.746	4.184

TABLE 4
Comparison of MSE to Groundtruth for SH Lighting Prediction From Various Methods

Method	MSE (10^{-2})
DeepLight	3.36 ± 0.29
Gardner *et al.*	6.59 ± 0.91
Neural Illumination	3.21 ± 0.80
Ours (trained without support plane)	7.54 ± 0.97
Ours	$\mathbf{2.29 \pm 0.11}$

by minor errors in the estimation of albedo, roughness, and lighting; therefore, achieving good photorealism.

We compare our method to DeepLight [21] and Gardner *et al.* [10], both of which take a single image as input. We use the origin author implementations. We evaluate DeepLight [21] by running the ARCore's illumination estimation and Gardner *et al.* [10] by running their web demo. We also compare our method with the NeuralIllumination [11] that is re-implemented by us and trained on the Matterport3D dataset.

For the quantitative comparison results in Table 4, 4,950 testing images are selected from Matterport3D and our collected outdoor data (these images are not in the training set). A testing sample is an {input image, ground-truth SH lighting} pair. For methods that output HDR environment maps, we will perform an SH fitting to extract the 5th-order SH coefficients before the quantitative comparison. The MSEs between the predictions by various methods and the groundtruth SH lighting are reported in Table 4. Our method has the lowest MSE.

In Fig. 11, we show results of object insertion. The images are selected from test sets of Matterport3D and our outdoor dataset. For each inserted virtual object, we compare the pixel color on the rendered image to the groundtruth rendering. It shows our method can generate good results for both indoor and outdoor input images.

We also compare our method with that of Garon *et al.* [22], which takes a single image as input and outputs *per-pixel* 5th-order SH coefficients. In Table 5, we compare the relighting error with Garon *et al.* [22] for "center" insertion and "off-center" insertion. The virtual object is a white Lambertian sphere. For center insertion, we render the sphere at the center of the image with predicted SH lighting and then calculate the rendered pixel color error with images rendered with groundtruth lighting. For off-center insertion, since Garon *et al.* [22] do not specify how far it is from the center, we insert the sphere to the position where it is $0.4\times$width-of-image to the center. The results are reported in Table 5. Our method outperforms Garon *et al.* [22] for center insertion cases around the foreground object due to better understanding of the scene. But for off-center insertion towards the image boundary, Garon *et al.* [22] is better due to per-pixel estimation of spatially varying lighting.

We also compare our methods with that of Chalmers *et al.* [25], which takes a single image as input and outputs reflection maps (RMs) from several roughness levels. In Fig. 12, we compare the object insertion results by using three different roughness levels. Our method has comparable results on high roughness levels (0.3 and 0.7). However, with low roughness level (0.1), StackedCNN[25] performs better because the predicted RM is not parameterized in any form and can potentially provide more details.

In Fig. 13, we provide results of virtual objects inserted into Internet photos not in our dataset. It shows the robustness of our approach, even for images that are not from Matterport3D or our collected outdoor data.

4.3 Mobile AR Application

To support real-time mobile AR applications, the total time spent on inference and rendering for one frame has to be less than 33ms to achieve 30fps. We will discuss our optimization strategies for light estimation and rendering.

4.3.1 Model Optimization

We have ported the entire end-to-end light estimation part in the Scene Parsing Network (Eqn. (2)) to Qualcomm's Snapdragon Hexagon DSP for a real-time solution. Hexagon DSP is an efficient low power processor. The inference model is first linearly quantized into an 8-bit fixed point model offline, then is run on the Hexagon DSP core with full HVX-SIMD optimization.

We tested the model both on a laptop CPU and mobile platforms. The results in Table 6 show the light estimation model runs at 1 fps on the laptop CPU (Intel Core i7-8750 @2.2 GHz CPU), 42 fps on the Qualcomm SD845, and 58 fps on Qualcomm SD855 devices (240x240 input image).

4.3.2 Real-Time Rendering on Mobile Device

Real-time rendering of shadows using SH lighting is complicated. Pre-computed Radiance Transfer (PRT) [51] can be used to achieve real-time soft shadows, but only with static scenes. Due to these limitations, in our mobile renderer, the SH lighting is only used for shading and the shadows are generated with the main directional light.

We use the microfacet BRDF model for all our virtual objects; therefore, similar to Section 3.5, the shading of virtual objects can be calculated via a low cost linear combination of polynomials. In our implementation, we designate the computation onto OpenGL's fragment shader. The rendering time for one frame is 5ms on average.

The total time for light estimation and rendering of a single frame is 28.8ms on the Qualcomm SD845 and 22.2ms on the Qualcomm SD855. Both of them can provide $>$ 30fps frame rate for real-time AR applications. In Fig. 14, we show some real-time virtual object rendering results in different real environments. For the complete demo video, please see the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2022.3141943.

4.4 Other Rendering Applications

In addition to lighting parameters, our approach also estimates the albedo, roughness, and normal map of a foreground object. With these information, rendering effects such as relighting and interreflection can be enabled to some extent. For relighting a real object, we use the estimated albedo, roughness, and normal of the foreground object. Fig. 15 shows the relighting results on a real object. In the relit images (second to fifth image), the novel lighting

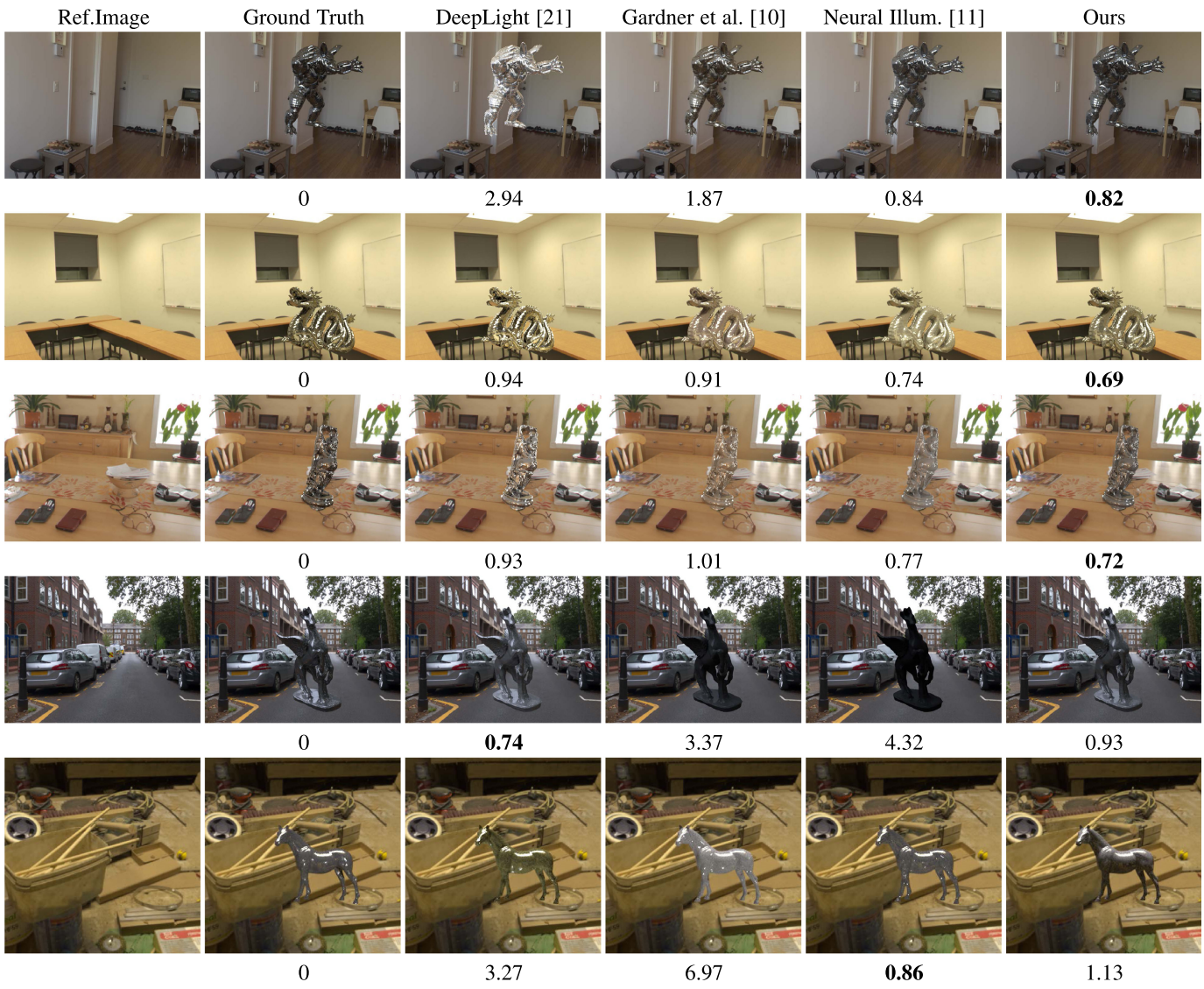


Fig. 11. Images with inserted relit virtual objects. The first column displays the input images. We display quantitative metrics (MSE with 10^{-1} as the unit) for the pixel color error of the rendered object underneath each result. In most of the cases, our method gives smaller MSE than other methods. Note that we selected four examples with low MSEs to display in the first four rows. After fitting, SH parameters among all methods are close to each other; thus, the rendering results are similar. In the final row, we show one example with higher MSE.

directions are from left, right, top, and bottom respectively. Qualitatively, the appearances on the relit object can correctly indicate the lighting direction.

For interreflection, we use Mitsuba [36] to visualize the interaction between the support plane and virtual object. We assume the surface of the support plane is Lambertian. Fig. 16 shows the color bleeding between virtual object and the real table surface. Other effects, such as casting shadow, rely on recovering the geometry of the scene. Although this can be done by using estimated normal maps and sparse geometry information provided by the AR framework using a method similar to ones in [52], [53], it is beyond the scope of this paper.

TABLE 5
Comparison of Relighting Error With Garon *et al.* [22]

		Center	Off-center
RMSE	Garon *et al.* [22].	0.049 ± 0.019	0.051 ± 0.012
	Ours	0.042 ± 0.017	0.072 ± 0.019
siRMSE	Garon *et al.* [22].	0.072 ± 0.011	0.055 ± 0.009
	Ours	0.067 ± 0.009	0.061 ± 0.013

We report Root Mean Square Error (RMSE) and scale-invariant Root Mean Square Error (siRMSE) [50].

5 Limitations and Future Work

One limitation is that the screen-space rendering algorithm described in this paper cannot effectively handle global illumination effects such as shadows, reflections, and refractions. Hence, the prediction accuracy of our method will degenerate on input images with these effects. For scenes in a low-light environment, our algorithm will not work well since the low-light signal is not sufficient for a good normal and BRDF prediction. Currently our method does not support spatially varying lighting.

In the future, we will extend the screen-space renderer to support screen-space global illumination (SSGI) [54]. In some game engines, such as Unreal, SSGI is a standard

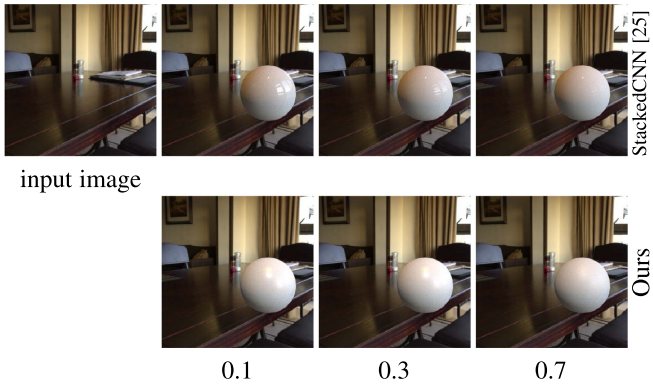


Fig. 12. Object insertion at three roughness level. The first row shows the results from StackedCNN [25] and the second row shows our results.

Fig. 13. Object relighting on a variety of generic stock photos downloaded from the Internet.

TABLE 6
Inference Time on Different Devices

Device	Inference time
CPU (Intel Core i7-8750 @2.2GHz)	970ms
Qualcomm SD845	23.8ms
Qualcomm SD855	17.2ms

Fig. 14. Our mobile AR application. In all images, the virtual bunny has shading and shadows blended with its surrounding environment seamlessly.

Fig. 15. Relighting on the real object. In the re-lit images (second to fifth), the novel lighting directions are from left, right, top, and bottom respectively.

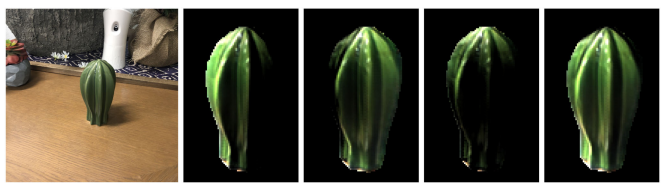


Fig. 16. The color bleeding between the virtual sphere the real desktop surface.

process to improve rendering quality. We will investigate how to parameterize the integrals in SSGI and make it differentiable to be embedded into a neural network. We will also consider extending to use cases that have less assumptions on the scene. More effective data collection and more effective physically-based network design will be the two major directions for achieving this goal.

6 Conclusion

We propose an approach for estimating the lighting condition from a single image in an AR scene. We also propose a low cost screen-space renderer to train a scene parsing network effectively. The training data contains both indoor and outdoor images, The indoor dataset was created from the publicly available Matterport3D. We extracted the groundtruth lighting and normal from the Matterport3D data for images with a support plane. The outdoor dataset was captured by using a rig similar to that of DeepLight [21]. We reconstructed the groundtruth lighting from captured images and used the outdoor dataset to fine-tune our model.

We show that utilizing the support plane in our network can improve the normal estimation and provide a good understanding of the 3D geometry. The screen-space renderer can be effectively differentiated, enabling end-to-end training. Finally, we show experimental results on benchmark data and real images captured on a mobile phone. Our method can estimate normal and lighting accurately under different lighting conditions. The rendered virtual objects can be blended into the real scene.

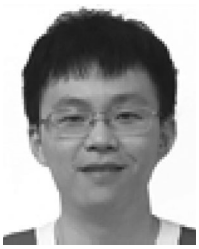
Acknowledgments

The work of Celong Liu was done when he was with OPPO.

References

[1] J. T. Kajiya, "The rendering equation," *ACM SIGGRAPH Comput. Graph.*, vol. 20, pp. 143–150, 1986.
[2] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Techn.*, 2008, pp. 189–198.
[3] H. Weber, D. Prévost, and J. Lalonde, "Learning to estimate indoor lighting from 3D objects," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 199–207.
[4] J. T. Barron and J. Malik, "Intrinsic scene properties from a single RGB-D image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 17–24.
[5] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner, "Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3133–3141.
[6] L. Gruber, T. Richter-Trummer, and D. Schmalstieg, "Real-time photometric registration from arbitrary geometry," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2012, pp. 119–128.
[7] R. Monroy, M. Hudon, and A. Smolic, "Dynamic environment mapping for augmented reality applications on mobile devices," in *Proc. Conf. Vis. Model. Vis.*, 2018, pp. 21–28.
[8] E. Zhang, M. F. Cohen, and B. Curless, "Emptying, refurnishing, and relighting indoor spaces," *ACM Trans. Graph.*, vol. 35, no. 6, 2016, Art. no. 174.
[9] E. Zhang, M. F. Cohen, and B. Curless, "Discovering point lights with intensity distance fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6635–6643.
[10] M.-A. Gardner *et al.*, "Learning to predict indoor illumination from a single image," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–14, 2017.
[11] S. Song and T. Funkhouser, "Neural illumination: Lighting prediction for indoor environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6911–6919.
[12] H. Weber, D. Prévost, and J.-F. Lalonde, "Learning to estimate indoor lighting from 3D objects," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 199–207.
[13] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem, "Rendering synthetic objects into legacy photographs," *ACM Trans. Graph.*, vol. 30, 2011, Art. no. 157.
[14] S. Georgoulis, K. Rematas, T. Ritschel, M. Fritz, T. Tuytelaars, and L. Van Gool, "What is around the camera?," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5180–5188.
[15] D. Mandl *et al.*, "Learning lightprobes for mixed reality illumination," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2017, pp. 82–89.
[16] J. Park, H. Park, S.-E. Yoon, and W. Woo, "Physically-inspired deep light estimation from a homogeneous-material object for mixed reality lighting," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 5, pp. 2002–2011, May 2020.
[17] D. A. Calian, J.-F. Lalonde, P. Gotardo, T. Simon, I. Matthews, and K. Mitchell, "From faces to outdoor light probes," *Comput. Graph. Forum*, vol. 37, pp. 51–61, 2018.
[18] S. B. Knorr and D. Kurz, "Real-time illumination estimation from faces for coherent rendering," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2014, pp. 113–122.
[19] C. LeGendre *et al.*, "Learning illumination from diverse portraits," in *Proc. SIGGRAPH Asia Tech. Commun.*, 2020, pp. 1–4.
[20] A. Chang *et al.*, "Matterport3D: Learning from RGB-D data in indoor environments," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 667–676.
[21] C. LeGendre *et al.*, "DeepLight: Learning illumination for unconstrained mobile mixed reality," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5911–5921.
[22] M. Garon, K. Sunkavalli, S. Hadap, N. Carr, and J.-F. Lalonde, "Fast spatially-varying indoor lighting estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6901–6910.
[23] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagne, and J.-F. Lalonde, "Deep parametric indoor lighting estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7174–7182.
[24] P. P. Srinivasan, B. Mildenhall, M. Tancik, J. T. Barron, R. Tucker, and N. Snavely, "Lighthouse: Predicting lighting volumes for spatially-coherent illumination," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8077–8086.
[25] A. Chalmers, J. Zhao, D. Medeiros, and T. Rhee, "Reconstructing reflection maps using a stacked-CNN for mixed reality rendering," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 10, pp. 4073–4084, Oct. 2021.
[26] M. Maximov, L. Leal-Taixé, M. Fritz, and T. Ritschel, "Deep appearance maps," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8728–8737.
[27] K. Rematas, T. Ritschel, M. Fritz, E. Gavves, and T. Tuytelaars, "Deep reflectance maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4508–4516.
[28] S. Georgoulis *et al.*, "Reflectance and natural illumination from single-material specular objects using deep learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1932–1947, Aug. 2018.
[29] Z. Li, Z. Xu, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Learning to reconstruct shape and spatially-varying reflectance from a single image," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–11, 2018.
[30] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and SVBRDF from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2472–2481.
[31] Y. Yu and W. A. P. Smith, "InverseRenderNet: Learning single image inverse rendering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3150–3159.
[32] D. Azinovic, T.-M. Li, A. Kaplanyan, and M. Nießner, "Inverse path tracing for joint material and lighting estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2442–2451.
[33] M. Janner, J. Wu, T. D. Kulkarni, I. Yildirim, and J. Tenenbaum, "Self-supervised intrinsic image decomposition," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5936–5946.
[34] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable Monte Carlo ray tracing through edge sampling," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 222:1–222:11, 2018.
[35] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3D reasoning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7707–7716.
[36] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *Trans. Graph.*, vol. 38, no. 6, Dec. 2019, Art. no. 203.
[37] C. Zhang, L. Wu, C. Zheng, I. Gkioulekas, R. Ramamoorthi, and S. Zhao, "A differential theory of radiative transfer," *ACM Trans. Graph.*, vol. 38, no. 6, 2019, Art. no. 227.
[38] A. Tewari *et al.*, "State of the art on neural rendering," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 701–727, 2020.

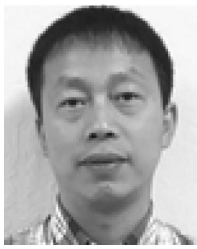
[39] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel, "Deep shading: Convolutional neural networks for screen space shading," *Comput. Graph. Forum*, vol. 36, no. 4, pp. 65–78, 2017.
[40] R. Ramamoorthi and P. Hanrahan, "A signal-processing framework for inverse rendering," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2001, pp. 117–128.
[41] L. Murmann, M. Gharbi, M. Aittala, and F. Durand, "A dataset of multi-illumination images in the wild," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4079–4088.
[42] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2
[43] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.
[44] B. Karis and E. Games, "Real shading in Unreal engine 4," *Proc. Physically Based Shading Theory Pract.*, vol. 4, pp. 3–8, 2013.
[45] B. Tunwattanapong *et al.*, "Acquiring reflectance and shape from continuous spherical harmonic illumination," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–12, 2013.
[46] Z. Li and N. Snavely, "Learning intrinsic image decomposition from watching the world," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9039–9048.
[47] P.-P. Sloan, "Stupid spherical harmonics (SH) tricks," in *Proc. Game Developers Conf.*, 2008, Art. no. 42.
[48] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *ACM Trans. Graph.*, vol. 38, no. 6, 2019, Art. no. 203.
[49] A. Shashua and T. Riklin-Raviv, "The quotient image: Class-based re-rendering and recognition with varying illuminations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 129–139, Feb. 2001.
[50] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, "Ground truth dataset and baseline evaluations for intrinsic image algorithms," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 2335–2342.
[51] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," in *Proc. 29th Annu. Conf. Comput. Graph. Interactive Techn.*, 2002, pp. 527–536.
[52] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combining positions and normals for precise 3D geometry," in *Proc. ACM SIGGRAPH Papers*, 2005, pp. 536–543.
[53] Y. Xu and D. G. Aliaga, "High-resolution modeling of moving and deforming objects using sparse geometric and dense photometric measurements," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1237–1244.
[54] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating dynamic global illumination in image space," in *Proc. Symp. Interactive 3D Graph. Games*, 2009, pp. 75–82.



Celong Liu received the bachelor's and master's degrees from Tsinghua University, Beijing, China, in 2011 and 2014, and the PhD degree from Louisiana State University, Baton Rouge, Louisiana, in 2019. His research interest lies in applying deep learning methods to solve computer graphics and computer vision problems.

Lingyu Wang received the master's degree from the University of Southern California, Los Angeles, California, in 2018. Her research interest include 3D computer graphics, especially physical-based rendering.

Zhong Li received the MSc degree in computer science from the University of Missouri, Columbia, Missouri, in 2015, and the PhD degree in computer science from the University of Delaware, Newark, Delaware, in 2019. He is currently a research scientist with InnoPeak Technology (OPPO US Research Center). His research interests include computational photography, computer graphics, and computer vision.



Shuxue Quan received the PhD degree from the Rochester Institute of Technology, Rochester, New York, in 2002. He is currently a senior research director of OPPO US Research Center, InnoPeak Technology Inc. His experiences cover camera/ISP, CV, AR, and AI. Before joining InnoPeak, he has taken industry positions with Sony, MICRO, Broadcom, and Qualcomm.

Yi Xu received the PhD degree from Purdue University, West Lafayette, Indiana, in 2010. He is currently the director of research (XR) with OPPO US Research Center, InnoPeak Technology Inc. His research interest lies in 3D computer graphics and computer vision, with a focus on augmented reality. Before joining InnoPeak, he worked at various industrial labs such as GE Research and JD.COM Silicon Valley Labs.