Supplementary Material
Physically-Based Editing of Indoor Scene Lighting from a Single Image

Zhengqin Li[1], Jia Shi[1,3], Sai Bi[1,2], Rui Zhu[1], Kalyan Sunkavalli[2], Miloš Hašan[2], Zexiang Xu[2], Ravi Ramamoorthi[1], and Manmohan Chandraker[1]

[1]UC San Diego [2]Adobe Research [3]Carnegie Mellon University

The supplementary material is organized as follows:

- A video to demonstrate the consistency of our scene editing results (Sec. 1)
- Network and training details (Sec. 2)
- More qualitative and quantitative light source prediction and neural rendering results on our synthetic dataset (Sec. 3)
- Light editing results with predicted masks (Sec. 4).
- Limitations and future works (Sec. 5).

1 Video

We include a video covering various scene editing applications. In the video, we move virtual light sources and objects to change highlights and shadows, and gradually modify the wall color to edit global illumination. Even though we do not explicitly add any smoothness constraint, our framework manages to achieve consistent scene editing results. This is probably because our framework explicitly follows the physics of the image formation process, which provides a natural regularization for our rendering results to be consistent.

2 Network Architecture and Training Details

We train our network on the OpenRooms Dataset [4], which is the only dataset that provides all the necessary ground truths for training our light source estimation and neural rendering frameworks, including depth $\mathbf{D}$, SVBRDF (diffuse albedo $\mathbf{A}$, normal $\mathbf{N}$ and roughness $\mathbf{R}$), direct shading $\mathbf{E}_{\mathcal{L},\mathcal{W}}$ and shadow $\mathbf{S}_{\mathcal{L},\mathcal{W}}$ for each individual light source, shading with indirect illumination $\mathbf{E}$ and per-pixel lighting $\mathbf{L}$. The dataset contains 1287 indoor scenes, with 118,233 images in total. We utilize 108,159 images rendered from 1178 scenes for training and the rest for testing. All the synthetic results are generated from the testing set. The comprehensive supervision provided by the OpenRooms dataset [4] allows us to train each module of our framework separately, which greatly simplifies the training process. The number of iterations for training each network and batch sizes are summarized in Tab. 3.

For all network figures in the supplementary, CX_1-KX_2-SX_3-PX_4-GX_5 represents a convolutional layer with X_1 channels, kernel size X_2, stride size X_3,

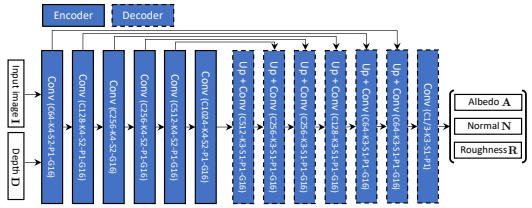

Fig. 1. Network architecture of **MNet** for material parameter prediction.

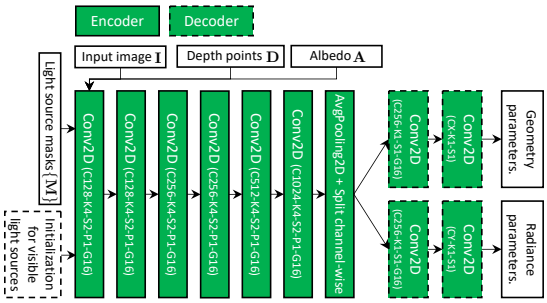

Fig. 2. Network architecture for light source prediction. All four light source prediction networks share a similar architecture. Networks for visible light source prediction have initialization of light source geometry as inputs. The numbers of output channels differ according to the light source type.

padding size X_4, followed by a group normalization layer with X_5 channels per group. Up represents a bilinear interpolation layer that doubles the resolution of the input feature map.

2.1 Material Prediction

We first train **MNet** for SVBRDF prediction. The network architecture is shown in Fig. 1. The input to the network is a 240×320 LDR image $\mathbf{I}$ and the depth map $\mathbf{D}$, while the outputs are diffuse albedo $\mathbf{A}$, normal $\mathbf{N}$ and roughness $\mathbf{R}$. Similar to prior work [3], we use three decoders but one shared encoder to predict three BRDF parameters because these parameters are correlated. We add skip-links to help reconstruct details. The loss function is the sum of L_2 loss on the three BRDF parameters.

$$\sum_{\mathbf{X} \in \{\mathbf{A}, \mathbf{N}, \mathbf{R}\}} (\mathbf{X} - \bar{\mathbf{X}})^2. \tag{1}$$

Note that we normalize the ground-truth $\mathbf{D}$ so that its mean is equal to 3 before we send it to every network. The reason is that there is a scale ambiguity for single image depth prediction using DPT network [6]. We find that this is important for the networks to generalize well to real images.

2.2 Light Source Prediction

Once we finish training the **MNet**, we use its predictions $\mathbf{A}$, $\mathbf{N}$ and $\mathbf{R}$ as inputs, combined with image $\mathbf{I}$, depth map $\mathbf{D}$ and light source segmentation mask $\mathbf{M_j}$ to train our light source prediction networks. We have four types of light source prediction networks for four types of light sources, {visible/invisible} and {lamp/window}, which share similar architectures but differ in parameterization

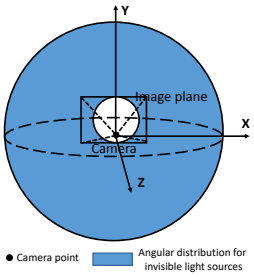

Fig. 3. Visualization of parameterization for the centers of invisible light sources. Our parameterization encourages centers of invisible light sources to be outside the camera frustum.

	sun	sky	ground
$(\lambda^{\min}, \lambda^{\max})$	(0.9, 1-10^{-6})	(0, 1-10^{-4})	(0, 1-10^{-4})

Table 1. The value of $\lambda^{\min}$ and $\lambda^{\max}$ for SGs correspond to sun, sky and ground respectively. $\lambda^{\max}$ is set slightly less than 1.

and inputs. As shown in Fig. 2, the encoder architecture consists of six 2D convolutional layers with stride 2. We also tried to use pre-trained ResNet-18 [2] but the results were worse.

Invisible light sources prediction The inputs to invisible window and invisible lamp prediction networks are exactly the same, which include the LDR image ($\mathbf{I}$), depth ($\mathbf{D}$), albedo ($\mathbf{A}$) and the sum of light source masks $\mathbf{M} = \sum_{\mathbf{j}} \mathbf{M_j}$. We first project 1-channel depth map $\mathbf{D}$ into 3-channel point cloud before we concatenate it with other inputs. The radiance of a lamp is controlled by intensity $\mathbf{w}$ only. We use tan function to project the initial network output $\tilde{\mathbf{w}}$ in range $[0, 1]$ to high dynamic range

$$\mathbf{w} = \tan(\frac{\pi}{2}\tilde{\mathbf{w}}). \tag{2}$$

The radiance of a window is controlled by 3 SGs, which correspond to sun, sky and ground of outdoor illumination. We encourage the SG corresponding to sun to represent high-frequency directional lighting. Therefore, we introduce two more parameters $\lambda^{\max}$ and $\lambda^{\min}$ for each SG to constrain their bandwidth parameters. Similarly, let $\tilde{\mathbf{w}}_\mathbf{k}$, $\tilde{\mathbf{d}}_\mathbf{k}$ and $\tilde{\lambda}_\mathbf{k}$ be the initial predictions, where $\tilde{\mathbf{d}}_k$ is in the range of [-1, 1], $\tilde{\mathbf{w}}_\mathbf{k}$ and $\tilde{\lambda}_k$ are in the range of $[0, 1]$, we have

$$\mathbf{w_k} = \tan(\frac{\pi}{2}\tilde{\mathbf{w}}_\mathbf{k}) \tag{3}$$

$$\mathbf{d_k} = \texttt{normalize}(\tilde{\mathbf{d}}_\mathbf{k}) \tag{4}$$

$$\lambda_\mathbf{k} = \tan(\frac{\pi}{2}(\tilde{\lambda}_\mathbf{k}(\lambda^{\max}_\mathbf{k} - \lambda^{\min}_\mathbf{k}) + \lambda^{\min}_\mathbf{k})) \tag{5}$$

The value of $\lambda^{\max}$ and $\lambda^{\min}$ parameters for sun, sky and ground are summarized in Tab. 1.

We represent the geometry of the invisible window and invisible lamp using a plane $\{\mathbf{c}, \mathbf{x}, \mathbf{y}\}$ and a 3D box $\{\mathbf{c}, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$ respectively, where $\mathbf{c}$ is the center and $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ are the three axes. To recover axes for lamps, we predict Euler angles α, β and γ as well as the axis length $l_\mathbf{x}$, $l_\mathbf{y}$ and $l_\mathbf{z}$, which combined together can be used to compute $\mathbf{x}, \mathbf{y}, \mathbf{z}$. To recover axes for windows, we first predict initial

axes $\tilde{\mathbf{y}}$ and $\mathbf{z}$ that are perpendicular to each other, as well as axis lengths $l_{\mathbf{x}}$ and $l_{\mathbf{y}}$. Let $\mathbf{u} = [0, 1, 0]$ be the up vector. The final axis predictions are computed as

$$\mathbf{y} = \texttt{normalize}(\tilde{\mathbf{y}} + \mathbf{u}) l_{\mathbf{y}} \tag{6}$$

$$\mathbf{x} = \texttt{normalize}(\texttt{cross}(\mathbf{z}, \mathbf{y})) l_{\mathbf{x}} \tag{7}$$

The intuition of introducing $\mathbf{u}$ is that for most windows, their $\mathbf{y}$ is close to up vector $\mathbf{u}$. Therefore, we predict the difference between $\mathbf{y}$ and $\mathbf{u}$ to make the training easier.

To predict centers of invisible light sources, we notice that if we directly output center $\mathbf{c}$, in some cases, the invisible light sources will be located inside the camera frustum or even across scene geometry, causing artifacts in the rendering. Thus, we change the parameterization to push their centers outside the camera frustum, as visualized in Fig. 3. We decompose the center $\mathbf{c}$ into direction $\mathbf{d_c}$ and length $l_{\mathbf{c}}$. Let f be the field of view for the short axis of the image plane, and $\mathbf{x_c}$, $\mathbf{y_c}$ and $\mathbf{z_c}$ be the camera coordinate system as shown in Fig. 3. We first predict $\theta_{\mathbf{c}} \in [0, \pi - f]$, $\phi_{\mathbf{c}} \in [-\pi, \pi]$ and $l_{\mathbf{c}} \in [0, \infty]$. Then, we compute center $\mathbf{c}$:

$$\mathbf{d_c} = \mathbf{x_c} \sin\theta_{\mathbf{c}} \cos\phi_{\mathbf{c}} + \mathbf{y_c} \sin\theta_{\mathbf{c}} \cos\phi_{\mathbf{c}} \tag{8}$$

$$+ \quad \mathbf{z_c} \cos\theta_{\mathbf{c}} \tag{9}$$

$$\mathbf{c} = \mathbf{d_c} l_{\mathbf{c}} \tag{10}$$

Visible window prediction Both geometry and radiance representations for visible windows are the same as those of invisible windows. However, for visible windows, instead of directly predicting their location from a single image, we first compute their initial light source center $\mathbf{c^{init}}$ based on their segmentation masks $\mathbf{M}_{\mathcal{W}}$ and depth $\mathbf{D}$, send the initial results to the network and then predict the difference between the initial estimation and the ground-truths. More specifically, we define $[\mathbf{X}; \mathbf{Y}; -\mathbf{1}]$ to be a 3-channel image representing pixel locations on the image plane. We introduce function $\mathbf{Edge}(\mathbf{M}, n)$ to compute the edge pixels of mask $\mathbf{M}$, where

$$\mathbf{Edge}(\mathbf{M}, n) = \texttt{dilation}(\mathbf{M}, n) - \mathbf{M}. \tag{11}$$

Formally, the initial center $\mathbf{c^{init}}$ is defined as

$$\mathbf{c^{init}} = \texttt{mean}([\mathbf{X}; \mathbf{Y}; -\mathbf{1}]\mathbf{M}_{\mathcal{W}}, \texttt{axis} = 3)$$
$$\texttt{mean}(\mathbf{DEdge}(\mathbf{M}_{\mathcal{W}}, 7))$$

$\mathbf{c^{init}}$ is then sent to the network by concatenating with other inputs as an extra 3-channel image. Let $\tilde{\mathbf{c}}$ be the output from the network. The final prediction $\mathbf{c}$ is defined as

$$\mathbf{c} = \mathbf{c^{init}} + \tilde{\mathbf{c}} \tag{12}$$

Visible lamp prediction The radiance of a visible lamp is simply represented by intensity $\mathbf{w}$. As shown in Fig. 5 in the main paper, the geometry of a visible lamp is represented by reflecting the visible surface with respect to the center to

compute the invisible and boundary area. Similar to the visible window case, we first compute the initial center as

$$\begin{aligned}\mathbf{c}^{\mathbf{init}} = &\texttt{mean}([\mathbf{X};\mathbf{Y};-\mathbf{1}]\mathbf{M}_{\mathcal{W}}, \texttt{axis} = 3)\\ &\texttt{mean}(\mathbf{D}\mathbf{M}_{\mathcal{W}})\end{aligned}$$

Since lamps are usually small and errors in their geometry prediction can cause highlight artifacts, we add a stronger regularization by requiring that the final predicted center $\mathbf{c}$ must be in the same camera ray of the initial center $\mathbf{c}^{\mathbf{init}}$. We decompose $\mathbf{c}^{\mathbf{init}}$ into direction $\mathbf{d}_{\mathbf{c}}^{\mathbf{init}}$ and length $l_{\mathbf{c}}^{\mathbf{init}}$ and predict $\tilde{l}_{\mathbf{c}}$ so that we have

$$\mathbf{c} = \mathbf{d}_{\mathbf{c}}^{\mathbf{init}}(l_{\mathbf{c}}^{\mathbf{init}} + \tilde{l}_{\mathbf{c}}) \tag{13}$$

Once we get the center $\mathbf{c}$, we can compute invisible area and boundary area based on that. Let $\mathbf{q}$ be one point on the visible area of a lamp, and $\mathbf{N}(\mathbf{q})$ be its normal. Let H be the height of the image. Recall that f is the field of view for the $\mathbf{y}$ axis. The area of $\mathbf{p}$ can be computed as

$$\texttt{area}(\mathbf{q}) = \left(\frac{1}{H}\tan(\frac{f}{2})\right)^2 \frac{1}{\max(\mathbf{N}(\mathbf{q})^T \cdot [0;0;1], 0)} \tag{14}$$

The corresponding invisible area $\hat{\mathbf{q}}$ can be computed as

$$\hat{\mathbf{q}} = 2(\mathbf{c} - (\mathbf{q}\cdot\mathbf{d}_{\mathbf{c}})\mathbf{d}_{\mathbf{c}}) + \mathbf{q} \tag{15}$$

$$\texttt{area}(\hat{\mathbf{q}}) = \texttt{area}(\mathbf{q}) \tag{16}$$

$$\mathbf{N}(\hat{\mathbf{q}}) = \mathbf{N}(\mathbf{q}) - 2(\mathbf{N}(\mathbf{q})\cdot\mathbf{d}_{\mathbf{c}})\mathbf{d}_{\mathbf{c}} \tag{17}$$

As for edge pixels, they can be computed as

$$\mathbf{Edge}(\mathbf{M}_{\mathcal{L}}, -1) = \mathbf{M}_{\mathcal{L}} - \texttt{erosion}(\mathbf{M}_{\mathcal{L}}, 1) \tag{18}$$

We create edge surface with center $\mathbf{q}_{\mathbf{e}}$ so that

$$\mathbf{q}_{\mathbf{e}} = \frac{\mathbf{q}+\hat{\mathbf{q}}}{2} \tag{19}$$

$$\texttt{area}(\mathbf{q}_{\mathbf{e}}) = ||\mathbf{q}-\hat{\mathbf{q}}|| \left(\frac{1}{H}\tan(\frac{f}{2})\right) \tag{20}$$

$$\mathbf{N}(\mathbf{q}_{\mathbf{e}}) = \texttt{normalize}(\mathbf{q}_{\mathbf{e}} - \mathbf{c}) \tag{21}$$

Note that both $\hat{\mathbf{q}}$ and $\mathbf{q}_{\mathbf{e}}$ are differentiable with respect to predicted center $\mathbf{c}$, which allows us to supervise the center prediction using the chamfer distance loss, which will be discussed below.

Loss functions We train the networks for visible lamps prediction using two loss functions $\mathbf{Loss}_{\mathbf{ren}}^{\mathbf{j}}$ and $\mathbf{Loss}_{\mathbf{geo}}^{\mathbf{j}}$. The $\mathbf{Loss}_{\mathbf{ren}}^{\mathbf{j}}$ is the L_1 loss between the rendered direct shading and the ground-truth of light source $\mathbf{j}$

$$\mathbf{Loss}_{\mathbf{ren}}^{\mathbf{j}} = |\mathbf{E}_{\mathbf{j}} - \bar{\mathbf{E}}_{\mathbf{j}}|.$$

We also considered $\log L_2$ loss but find that it focuses too much on low intensity regions and can cause wrong highlights. The geometry loss has two parts, a

ω_{sun}	ω_{sky}	ω_{grd}	$\omega_{\mathbf{w}}$	$\omega_{\mathbf{d}}$	ω_{λ}
1.0	0.2	0.2	0.001	1.0	0.001

Table 2. Value of coefficients for direct light source loss $\mathbf{Loss_{src}}$.

RMSE Chamfer distance loss and L_1 area loss. To compute the RMSE Chamfer distance loss, we randomly sample points on the surface of lamps and compute their RMSE Chamfer distance between the points sampled from ground-truth geometry. We find that compared to standard L_2 Chamfer distance, RMSE Chamfer distance can make the training more stable, especially for invisible light sources. To compute the L_1 area loss, we compute the sum of surface area of the predicted lamp and the ground-truth lamp. We observe that the area loss is important in preventing the network from predicting too large light sources, which may cause shadows to be blurry.

$$\mathbf{Loss^{j}_{geo}} = \mathbf{Cham}(\{\mathbf{q_j}\}, \{\bar{\mathbf{q}}_{\mathbf{j}}\}) + \omega_a |\mathtt{area}(\mathbf{j}) - \mathtt{area}(\bar{\mathbf{j}})|,$$

where ω_a is equal to 0.8.

For visible windows, we use the same $\mathbf{Loss^{j}_{ren}}$ and $\mathbf{Loss^{j}_{geo}}$ but also add direct supervision on the light source spherical Gaussian parameters. The way we compute the ground-truth window radiance parameters is introduced in Sec. 2.7. In summary, we use L_2 loss to supervise direction $\mathbf{d}$ and $\log L_2$ loss to supervise intensity $\mathbf{w}$ and bandwidth λ.

$$\begin{aligned}\mathbf{Loss^{\mathcal{W}}_{src}} = \sum_{k}^{\text{sun, sky, grd}} \omega_{\mathbf{k}} \Big(& \omega_{\mathbf{w}} || \log(\mathbf{w_k}+1) - \log(\bar{\mathbf{w}}_{\mathbf{k}}+1) ||^2 \\ & + \omega_{\mathbf{d}} ||\mathbf{d_k} - \bar{\mathbf{d}}_{\mathbf{k}}||^2 + \omega_{\lambda} || \log(\lambda_k+1) - \log(\bar{\lambda}_k+1) ||^2 \Big)\end{aligned} \tag{22}$$

The values of coefficients $\omega_{\cdot}$ are summarized in Tab. 2, which are determined by first fine-tuning on a small training set and then applying to the whole dataset. For invisible windows and lamps, we use the same loss functions as in the visible cases. Note that we only predict one invisible lamp and one invisible window for each image. When there is no invisible window or lamp, we only compute the rendering loss by setting $\bar{\mathbf{E}}_{\mathbf{j}} = \mathbf{0}$. When there are more than one invisible windows or lamps, we pick up the one whose direct shading $\bar{\mathbf{E}}_{\mathbf{j}}$ has the highest total energy and compute all losses with respect to it.

2.3 Shadow prediction.

For shadow prediction, we train **DShdNet** to in-paint and denoise the incomplete shadow map due to the occlusion boundaries. The network architecture of **DShdNet** is shown in Fig. 4, where we use a light-weight encoder-decoder structure with skip-links. We train the shadow prediction network with scale-invariant gradient based loss as proposed in [5] and find that it works much better than standard L2 loss, especially on real images. Formally, let $\mathbf{S}$ and $\bar{\mathbf{S}}$ be

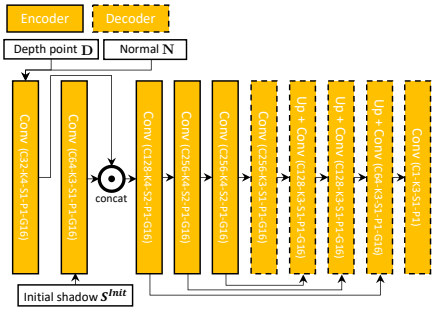

Fig. 4. Network architecture for shadow prediction.

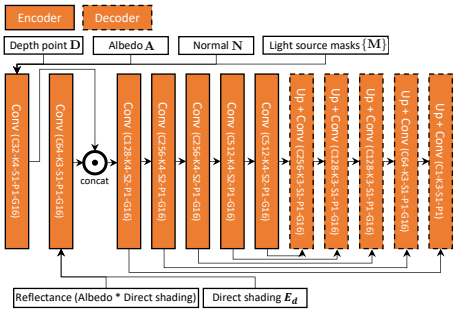

Fig. 5. Network architecture for indirect shading prediction.

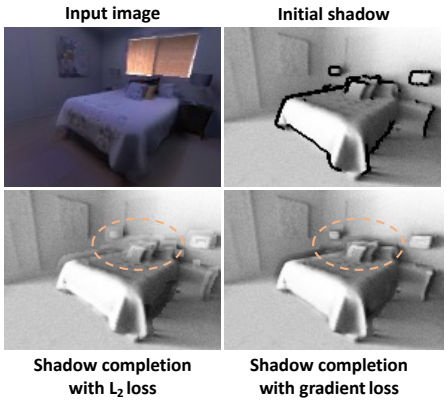

Fig. 6. Qualitative comparison of shadow completion network trained with L_2 loss and gradient-based loss. The orange circle highlights the artifacts in the result trained with L_2 loss.

the predicted and ground-truth shadows, the loss function is defined as

$$\mathbf{Loss_{shd}} = \sum_{h}^{1,2,4,8} \sum_{i,j} ||g_h[\mathbf{S}](i,j) - g_h[\bar{\mathbf{S}}](i,j)||^2,$$

where

$$g_h[\mathbf{S}](i,j) = \left(\frac{\mathbf{S}(i+h,j) - \mathbf{S}(i,j)}{|\mathbf{S}(i+h,j) + \mathbf{S}(i,j)|}, \frac{\mathbf{S}(i,j+h) - \mathbf{S}(i,j)}{|\mathbf{S}(i,j+h) + \mathbf{S}(i,j)|} \right).$$

Fig. 6 compares the shadow completion network trained with standard L_2 loss and the gradient-based loss on a real image from the Replica dataset [8]. We observe that compared to L_2 loss, gradient-based loss leads to smoother prediction with fewer artifacts.

2.4 Indirect illumination prediction

The network architecture for indirect shading prediction is shown in Fig. 5. which has an encoder-decoder architecture with a large receptive field so that the network can learn the non-local global information. We train this module

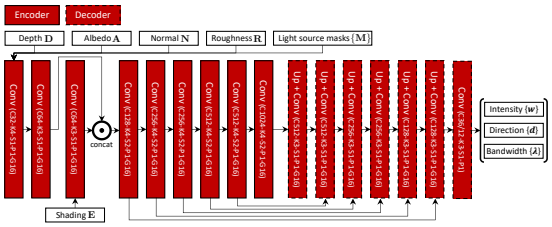

Fig. 7. Network architecture for lighting prediction.

	Mat.	Window		Lamp		Shadow	Indirect	Lighting
		Vis.	Inv.	Vis.	Inv.			
Iter(k)	135	120	200	120	150	70	180	240
Batch	12	9	9	9	9	3	3	3

Table 3. The number of iterations and batch size for training each network. For all networks, we use Adam optimizer with learning rate 10^{-4} and `betas` $= (0.9, 0.999)$.

independently by taking the ground-truth direct shading $\bar{\mathbf{E}}_\mathbf{d}$ as input and predict the indirect shading.

$$\mathbf{E_{ind}} = \mathbf{IndirectNet}(\bar{\mathbf{E}}_\mathbf{d}, \mathbf{D}, \mathbf{N}, \mathbf{A}) \tag{23}$$

$$\mathbf{E} = \mathbf{E_{ind}} + \bar{\mathbf{E}}_\mathbf{d} \tag{24}$$

The loss function is the L_1 loss of the final shading prediction $\mathbf{Loss_{shg}} = |\mathbf{E} - \bar{\mathbf{E}}|$.

2.5 Per-pixel lighting prediction

As shown in Fig. 7, the network architecture of **LightNet** for per-pixel lighting prediction is exactly the same as in [3]. The only difference is that we replace the input image $\mathbf{I}$ with the ground-truth per-pixel shading $\bar{\mathbf{E}}$.

$$\mathbf{L} = \mathbf{LightNet}(\bar{\mathbf{E}}, \mathbf{M}, \mathbf{A}, \mathbf{N}, \mathbf{R}, \mathbf{D}) \tag{25}$$

The loss function is the log L_2 loss on the predicted per-pixel lighting plus a L_1 shading loss. To compute the shading loss, we integrate the per-pixel lighting L to get shading prediction $\mathbf{E^L}$ and compute its difference from $\bar{\mathbf{E}}$.

$$\mathbf{Loss_{light}} = ||\log(\mathbf{L}+1) - \log(\bar{\mathbf{L}}+1)||^2 + \omega_r |\mathbf{E}^L - \bar{\mathbf{E}}|$$

where ω_r is set to be 0.01.

2.6 Differentiable Direct Shading Rendering Layer

We first discuss how we uniformly sample the surface of light sources. Let u, v be two random variables sampled from a uniform distribution in the range of $[-1, 1]$. To sample a point on the window surface, we have

$$\mathbf{q} = \mathbf{c} + 0.5u\mathbf{x} + 0.5v\mathbf{y} \tag{26}$$

The invisible lamp is represented by a 3D bounding box. Thus, we sample each of the 6 faces separately, for example,

$$\mathbf{q} = \mathbf{c} + 0.5\mathbf{x} + 0.5u\mathbf{y} + 0.5v\mathbf{z}. \tag{27}$$

For visible lamps, we treat every visible point $\mathbf{q}$, invisible point $\hat{\mathbf{q}}$ and edge point $\mathbf{q_e}$ as a plate whose normal $\mathbf{N}$ and area can be computed as in Eq. (14)-(21). Replacing these sampled points into Eq. (1) in the main paper allows us to compute the direct shading for every light source.

For window light sources specifically, we observe that it is necessary to sample according to the angular distribution of high-frequency directional sunlight (Fig. 4 in the main paper). To achieve this, we use standard Monte Carlo sampling by first computing the CDF of $\mathcal{G}_{\text{sun}} = \{\mathbf{w}_{\text{sun}}, \mathbf{d}_{\text{sun}}, \lambda_{\text{sun}}\}$ and then sampling the lighting direction using its inverse function. We define θ_{sun} and ϕ_{sun} to be the polar angle and azimuthal angle in a coordinate system where $\mathbf{d}_{\text{sun}}$ is the $\mathbf{z}$ axis. The PDF for θ_{sun} and ϕ_{sun} are

$$\begin{aligned}
\mathbf{Pr}(\phi_{\text{sun}}) &= \frac{1}{2\pi} \\
\mathbf{Pr}(\theta_{\text{sun}}) &= \frac{\lambda_{\text{sun}} \exp(\lambda_{\text{sun}}(\cos\theta_{\text{sun}} - 1)\sin\theta_{\text{sun}}}{1 - \exp(-2\lambda_{\text{sun}})} \\
\mathbf{Pr}(\phi_{\text{sun}}, \theta_{\text{sun}}) &= \mathbf{Pr}(\phi_{\text{sun}})\mathbf{Pr}(\theta_{\text{sun}})
\end{aligned}$$

Similar as when we sample the surface of light sources, let u, v be two random variables. To sample ϕ_{sun}, since it distributes uniformly, we simply have

$$\phi_{\text{sun}} = u\pi$$

To sample θ_{sun}, we first compute the CDF of its distribution

$$\mathbf{F}(\theta_{\text{sun}}) = \frac{\exp(\lambda_{\text{sun}}) - \exp(\lambda_{\text{sun}}\cos\theta_{\text{sun}})}{\exp(\lambda_{\text{sun}}) - \exp(-\lambda_{\text{sun}})},$$

and then compute θ_{sun} by inverting its CDF

$$\theta_{\text{sun}} = \arccos\left(\frac{\log(1 - \frac{v+1}{2}(1 - \exp(-2\lambda_{\text{sun}})))}{\lambda_{\text{sun}}} + 1\right)$$

Given θ_{sun} and ϕ_{sun}, the sampled lighting direction $\mathbf{l}$ can be computed as

$$\begin{aligned}
\mathbf{l} = \mathbf{x}_{\text{sun}} \sin\theta_{\text{sun}} \cos\phi_{\text{sun}} + \mathbf{y}_{\text{sun}} \sin\theta_{\text{sun}} \cos\phi_{\text{sun}} \\
+ \quad \mathbf{d}_{\text{sun}} \cos\theta_{\text{sun}}
\end{aligned}$$

which can be replaced into Eq. (2) in the main paper to compute the direct shading. Here $\mathbf{x}_{\text{sun}}$ and $\mathbf{y}_{\text{sun}}$ are two arbitrary orthogonal vectors perpendicular to $\mathbf{d}_{\text{sun}}$. Note that we implement all the sampling algorithms in pytorch so that our rendering layer is differentiable.

In Fig. 4 in the main paper and Fig. 8, we demonstrate that sampling according to both the geometry and radiance distribution of a window following the MIS rule can lead to much less noise with similar number of samples, compared to only uniformly sampling the surface area of the window.

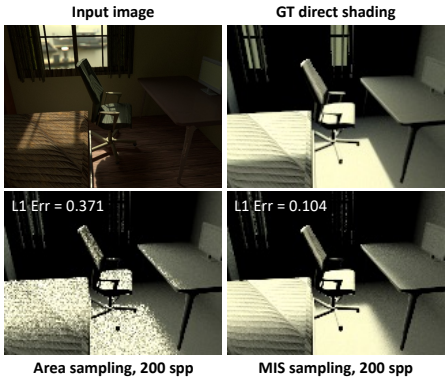

Fig. 8. Comparions of direct shading rendered by sampling area uniformly or using MIS sampling. Our MIS sampling has much less noise with the same number of samples. This makes it possible for us to train our networks with rendering loss, which is essential to achieve accurate light source reconstruction.

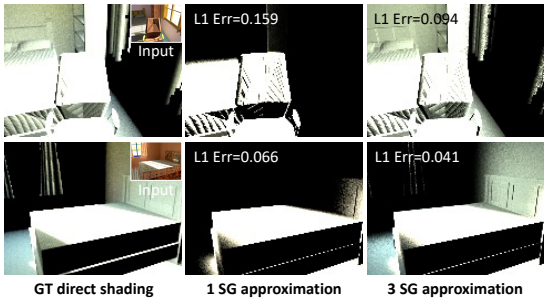

Fig. 9. A demonstration of direct shading rendered from our ground-truth window radiance parameters. Our ground-truth 3 SGs can be used to render direct shading that closely matches the ground-truth and is more expressive compared to a single SG representation, which cannot capture the ambient lighting.

2.7 Ground-truth Window Radiance Parameters

Our Monte Carlo-based differentiable direct shading rendering layer allows us to compute ground-truth radiance parameters for windows, by minimizing the rendering loss between the rendered direct shading $\mathbf{E}_{\mathcal{W}}$ and the ground-truth direct shading $\bar{\mathbf{E}}_{\mathcal{W}}$ provided by the OpenRooms dataset,

$$\mathbf{Loss_{ren}} = |\mathbf{E}_{\mathcal{W}} - \bar{\mathbf{E}}_{\mathcal{W}}| \tag{28}$$

Here we use L_1 loss instead of $\log L_2$ loss because we observe that the latter can recover low-intensity regions more accurately but meanwhile can lead to highlight artifacts. To encourage the 3 SGs to represent 3 physically meaningful light sources, sun, sky and ground respectively, we first render a panorama facing outside the window and then select the brightest direction in the panorama as the sunlight direction and keep it fixed through the optimization process. As for the other 2 SGs corresponding to sky and ground, we initialize their direction with up vector [0, 1, 0] and minus up vector [0, -1, 0] in the world coordinate system. In addition, we also apply the λ constraint as shown in Tab. 1 so that the high-frequency directional lighting can be mainly represented by the $\mathcal{G}_{\text{sun}}$.

In Fig. 4 in the main paper and Fig. 9, we demonstrate that our ground-truth 3 SGs parameters can be used to render direct shading very close to the ground-truth, with both high-frequency directional lighting and ambient lighting being correctly modeled, while the 1 SG representation applied by prior work [9] can only capture the directional lighting. Our ground-truth 3 SGs parameters are

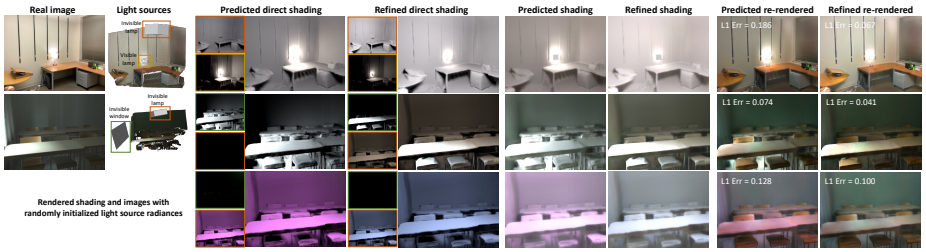

Fig. 10. Comparisons of light source prediction and re-rendered image before and after optimization. We observe that while the optimization-based refinement can help predict more consistent light source intensity, it also relies on a good initialization from the network to converge to a good result: a random initialization cannot lead to accurate recovery of light source radiance through pure optimization. This is especially true for more complex sunlight coming through windows. Note that the direct shading from the invisible window for the first example is always **0** and therefore is not shown here.

Material	Light source	Direct shading	Shadow	Indirect shading	Per-pixel lighting	Total
299ms	19.7ms	595ms	1309ms	19.1ms	19.35ms	2.26s

Table 4. Inference time of each step of our framework.

used to compute the $\mathbf{Loss}_{\mathbf{src}}^{\mathcal{W}}$ as shown in Eq. (22) in the training process and demonstrated to help capture more accurate and interpretable lighting in Sec. 3.

2.8 Optimization-based Refinement

Our differentiable rendering pipeline allows us to refine the light source radiance parameters based on rendering loss. We find that this is especially useful when the intensity of light source prediction can be slightly off sometimes. Given the light source parameters $\{\mathcal{W}\}$ and $\{\mathcal{L}\}$, which cover visible/invisible and windows/lamps, we can render shading $\mathbf{E}$. We define the rendering loss as the L_2 loss between the rendered image and the input LDR image,

$$\mathbf{Loss}_{\mathbf{ren}} = ||\min(\mathbf{EA}, 1) - \mathbf{I}||^2, \tag{29}$$

where $\mathbf{A}$ is the predicted albedo. Note that we have already transformed the input LDR image into linear RGB space. One alternative to compute the rendering loss is to use per-pixel lighting $\mathbf{L}$ so that we can also render specularity. However, we observe that it will cause the optimization to be unstable.

Fig. 8 and Fig. 10 compares the light source prediction and re-rendered image before and after optimization, where we observe that our rendering error-based optimiztion can effectively correct the intensity of the light source prediction. However, we also observe that for more complex sunlight coming through a window, it is important to provide a good initial prediction from the network. Otherwise, a pure optimization-based method cannot recover light source radiance correctly. In the second example in Fig. 10, we randomly initialize the light source radiance and observe that reconstructed direct shading and final re-rendered image may not be accurate.

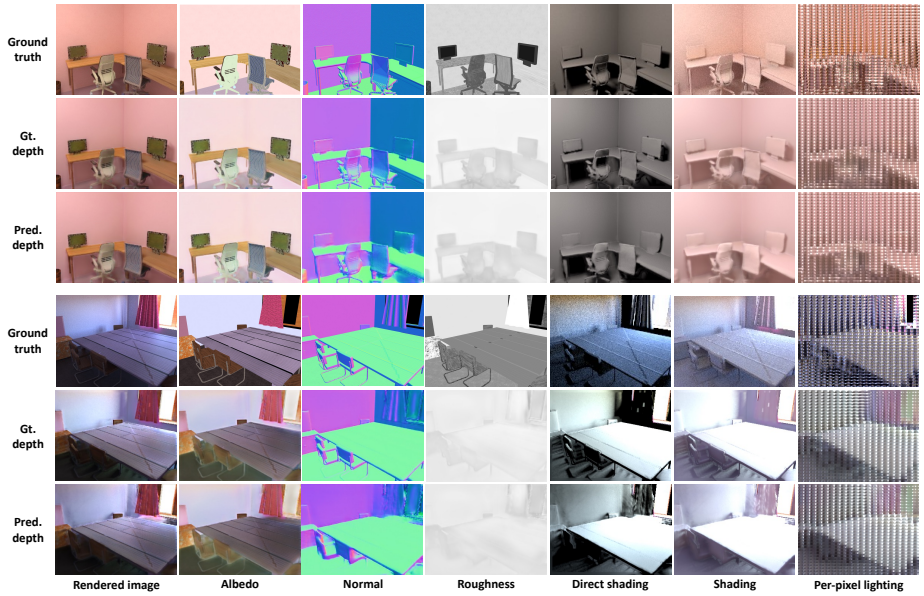

Fig. 11. Material predictions and neural rendering results on the OpenRooms synthetic dataset with predicted and ground-truth depth. We observe that with both ground truth our method can render high-quality direct shading, shading, per-pixel environment map and final image from our light source and material predictions, with non-local shadows and interreflection being correctly modeled.

2.9 Inference Time

The inference time for each step of the network to process one image is summarized in Tab. 4. The most time consuming step is to render shadows from depth using path tracing. Note that while our framework handles many complex light transport effects, including global illumination, the total time for it to reconstruct and re-render an indoor scene is only less than 3 s.

3 Synthetic Experiments on OpenRooms

We present more qualitative and quantitative results on the synthetic OpenRooms dataset [4]. More specifically, we test the effectiveness of different loss terms and how imperfect depth prediction can impact our light source prediction and neural rendering results. Our depth prediction are produced by DPT [6] without fine-tuning on our synthetic dataset. We train all our models on ground-truth depth and, as shown in multiple figures (e.g. Fig. 1, 12 and 13) in the main paper and Fig. 11 and 12, find that they generalize well to predicted depth for both real and synthetic data in most cases.

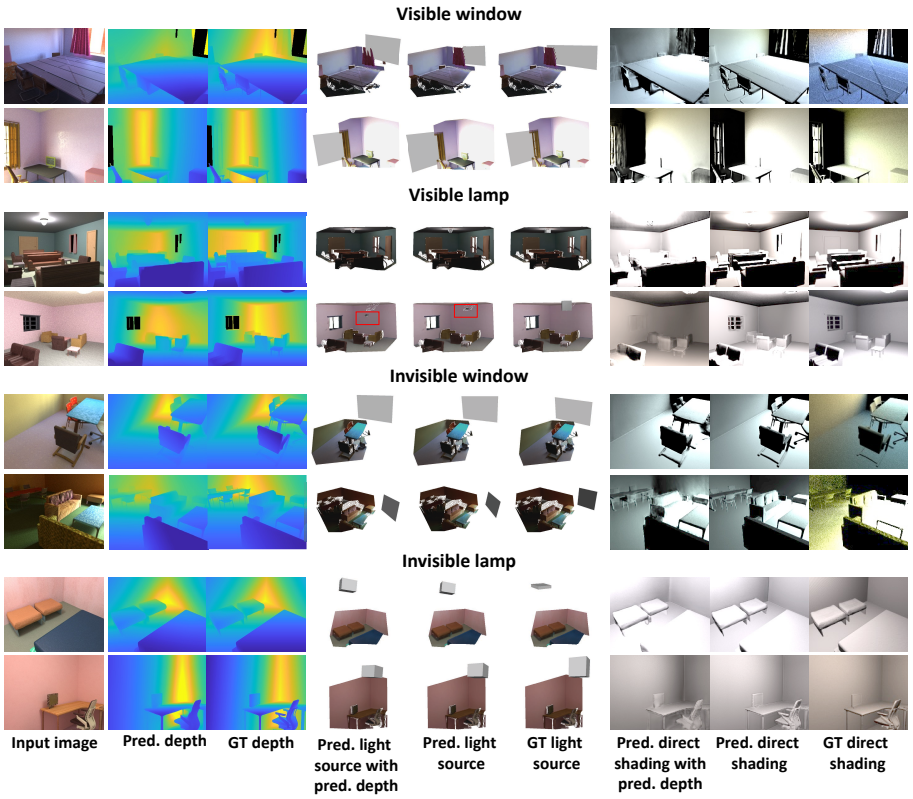

Fig. 12. Light source prediction results on the synthetic dataset for various types of light sources with ground-truth and predicted depth. In most cases, our method can recover both the geometry and radiance of light sources similar to ground truth with either predicted or ground-truth depth. We also show one example on the fourth row, as marked by the red rectangle, where the inaccurate depth prediction leads to poor geometry prediction of a visible lamp, causing the highlight in the shading to be missing.

3.1 Material prediction

Unlike the prior method [3], which first uses scale-invariant loss for albedo prediction and adopts a linear regression to solve the scale ambiguity, we use the absolute loss for both diffuse albedo and light intensity prediction. The reason is that our method needs to recover the radiance of multiple light sources in the scene and it is difficult to recover consistent intensities across multiple light sources through simple linear regression.

Tab. 5 compares our material prediction with [4]. We report the quantitative numbers with both ground-truth and predicted depth maps as inputs. When using ground-truth depth as an input, our normal prediction is much more accurate compared to [4]. Our roughness quality is similar to [3]. Both the roughness and albedo predictions are relatively insensitive to the depth accuracy. In Fig. 11 and Fig. 10 in the main paper, we present our material predictions on both

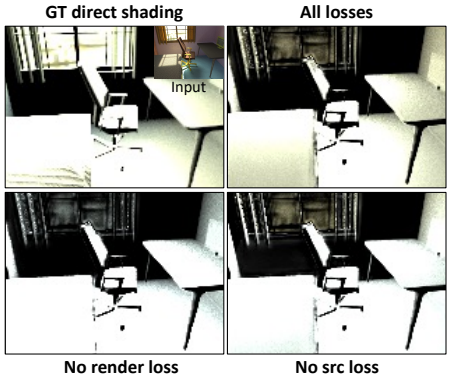

Fig. 13. Ablation studies on different loss combinations for window light source prediction. Our network trained with both rendering loss $\mathbf{Loss}^{\mathbf{j}}_{\mathbf{ren}}$ and light source loss $\mathbf{Loss}^{\mathbf{j}}_{\mathbf{src}}$ predicts the most accurate radiance, with both high-frequency directional lighting and ambient lighting closely matching the ground-truth.

	Albedo 10^{-2} **A**		Normal 10^{-2} **N**		Roughness 10^{-2} **R**	
Ours	Gt.	Pred.	Gt.	Pred.	Gt.	Pred.
	1.81	2.48	1.39	6.52	6.22	6.58
Li et al. [4]	-		4.51		6.59	

Table 5. Material predictions on the OpenRooms testing set. We report L_2 error of our material predictions. We report our results with both ground-truth depth and predicted depth as inputs. The network is trained with ground-truth depth and not fine-tuned with predicted depth.

real and synthetic data. On synthetic data, we show that our diffuse albedo, roughness and normal predictions are reasonably close to the ground truths. For real images, even though we do not have ground-truths, our material predictions are high-quality enough to enable realistic re-rendering of the scene.

3.2 Light source prediction

In Fig. 12, we show more qualitative light source prediction results using either ground-truth or predicted depth. The quantitative numbers are summarized in Tab. 3 in the main paper. In most cases, our light source prediction models, even though trained on ground-truth depth only, can generalize well to predicted depth and can recover geometry and radiance of all 4 types of light sources accurately. In the reconstructed direct shading, small errors can be seen caused by the imperfect depth prediction with less details, which might be inevitable. Our visible lamp reconstruction is more sensible to depth accuracy compared to other kinds of light sources due to its geometry representation. In the fourth row, we show one example where the inaccurate depth prediction causes the lamp position to be closer to the camera than the ground truth. Hence, the highlights on the floor is missing. This example may suggest that utilizing lighting information to improve geometry reconstruction can be an interesting future direction.

Ablation study Tab. 6 and Fig. 13 verifies the effectiveness of our loss functions for window light source prediction. We observe that while training with light source loss $\mathbf{Loss}^{\mathbf{j}}_{\mathbf{src}}$ can lead to the prediction closest to our optimized ground-truth

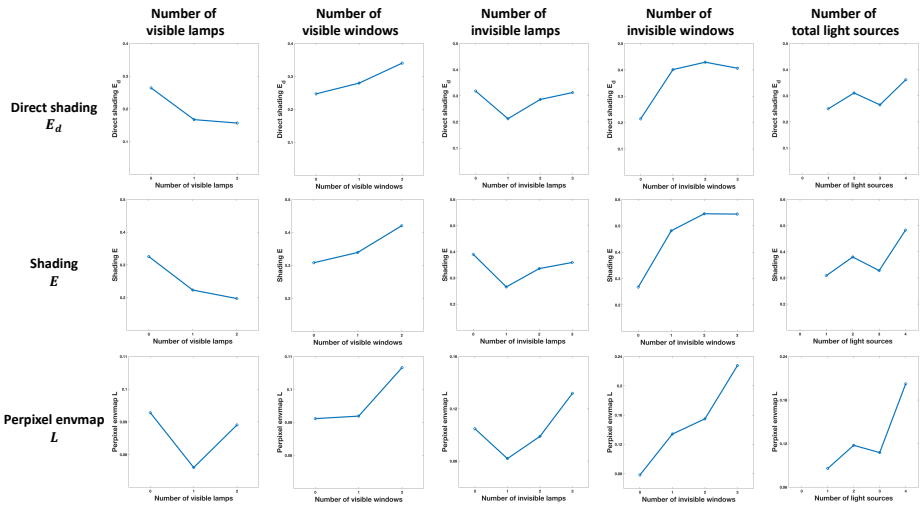

Fig. 14. Rendering error distribution with respect to the number of light sources. We observe that error increases when the number of windows increases, which is because the radiance of windows are more complex and difficult to predict.

light source parameters, the rendering error is significantly higher because it is difficult to find the best balance across different parameters that can minimize the rendering error. Training with $\mathbf{Loss}^{\mathbf{j}}_{\mathbf{ren}}$ alone leads to reasonable direct shading prediction. However, the light source parameters are less interpretable, as shown in Tab. 6 and the rendered direct shading tends to be oversmoothed, as shown in Fig 13. Combining the two losses together, on the contrary, allows us to render direct shading closer to the ground-truth, with high-frequency lighting being correctly modeled, as shown in both Tab. 6 and Fig. 13.

3.3 Neural rendering

In Fig. 11, we also show more neural rendering results with both predicted and ground-truth depth. Our physically-based neural rendering module is reasonably robust to depth inaccuracy, which can reconstruct high-quality direct shading, shading and per-pixel lighting similar to the ground truths.

Error distribution We report distribution of errors in Tab. 4 in the main paper with respect to the number of light sources in Fig. 14. Error increases when a scene has more windows or total number of light sources. It decreases or fluctuates with more lamps possibly because radiance of lamps can be predicted more accurately.

4 Light Editing with Predicted Masks

In all our prior synthetic and real experiments, we assume that the light source segmentation masks are given. While not being our focus, we fine-tuned a Mask

Visible window	Rendering	Light source		
	Direct	Intensity	Direction	Bandwidth
	$\mathbf{E_j}$	$\mathbf{w}$	$\mathbf{d}$	λ
w/o $\mathbf{Loss^j_{ren}}$	1.276	**7.972**	0.386	**4.369**
w/o $\mathbf{Loss^j_{src}}$	0.859	17.73	0.503	7.492
All	**0.849**	10.28	**0.369**	4.419
Invisible window	Rendering	Light source		
	Direct	Intensity	Direction	Bandwidth
	$\mathbf{E_j}$	$\mathbf{w}$	$\mathbf{d}$	λ
w/o $\mathbf{Loss^j_{ren}}$	1.786	**10.817**	0.545	**4.770**
w/o $\mathbf{Loss^j_{src}}$	0.334	44.04	1.432	70.48
All	**0.312**	18.15	**0.536**	8.168

Table 6. Ablation studies on window light source prediction. We report L_1 loss for direct shading E_j, L_2 loss for direction $\mathbf{d}$ and $\log L_2$ loss for intensity $\mathbf{w}$ and λ.

Fig. 15. Light source detection and instance segmentation results on the OpenRooms dataset [4].

RCNN [1] on the OpenRooms dataset and report its performances. The fine-tuned Mask RCNN can detect and segment 4 types of objects, windows and lamps, on and off. Quantitative and qualitative results are summarized in Tab. 7 and Fig. 15 respectively, where we observe our fine-tuned model works well on the synthetic dataset. This fine-tuned model can be used for real image editing by providing the initial light source segmentation masks, as will be discussed next.

We apply the fine-tuned MaskRCNN described above on a real image and see if an imperfect light source segmentation mask can still enable high-quality light editing applications. We first use our fine-tuned MaskRCNN to get an initial segmentation mask and then use the GrabCut method [7] to refine its boundaries. The results are summarized in Fig. 16. We observe that even though the mask prediction is not perfect, our light editing results are very similar to those shown in Fig. 12 in the main paper with a manually created mask, which suggests that our light source prediction and neural renderer can be robust to small mask prediction errors.

Fig. 16. Real image light editing results with predicted light source segmentation mask. The light editing results from a manually created mask are shown in the insets.

Metric/Type	bbox	seg
AP(0.5:0.95)	65.4	59.4
AR(0.5:0.95)	85.1	78.1
AP-windows-on	75.4	57.0
AP-lamp-on	70.4	72.1
AP-windows-off	54.0	50.0
AP-lamp-off	61.8	63.6

Table 7. Quantitative evaluation on bounding box regression and mask on OpenRooms[4] for light source (windows and lamps) detection and instance segmentation.

5 Limitations and Future Works

In this section, we analyze the limitations of our indoor light editing framework. We mainly focus on failure cases caused by our deliberate design choices to highlight the trade-offs being made to build our framework.

Non-symmetric lamps Our visible lamp representation assumes that lamps are symmetric with respect to their centers. While this simple assumption holds in many cases, especially for ceiling lamps, it can fail and cause highlight artifacts. Fig. 17 shows an example where the geometry of the lamp cannot be simply represented by reflecting its visible area. Our visible lamp representation will cause highlight artifacts projected on the wall in this example. The same artifact can be observed by comparing Fig. 1 (d) and (d.1) in the main paper, as in Fig. 1(d) our lamp model projects wrong highlights on the wall behind.

Separation of shading and visibility Our neural rendering framework separates visiibility ($\mathbf{S_j}$) and shading ($\mathbf{E_j}$) by assuming the direct shading $\mathbf{E_d}$ can be computed as

$$\mathbf{E_d} = \sum_{\mathbf{j}} \mathbf{S_j} \mathbf{E_j}. \tag{30}$$

There are two reasons we make this assumption. The first is that we hope to avoid checking the visibility for each ray in the rendering layer, which is too expensive and hard to be differentiable. The other is that we hope to introduce the shadow inpainting network **DShdNet**, which can handle artifacts caused by occlusion boundaries robustly and is necessary when we render shadow from a mesh created from a single depth map. While Eq. (30) works well for diffuse area lights, it may not work on directional light, where the visibility of each sampled ray should be considered separately. Fig. 18 compares $\mathbf{E_d}$ computed as in Eq. (30)

Fig. 17. A example where an indoor lamp is not symmetric and does not emit light uniformly in every direction.

Fig. 18. While our separation of shading and shadow is necessary for indoor scene light editing, it can cause missing details in direct shading $\mathbf{E_d}$, as shown in the green circle.

and the ground-truth direct shading. We can see that the ground-truth direct shading has more detailed highlight boundaries.

Missing geometry While our shadow inpainting network can handle artifacts caused by occlusion boundaries, it cannot handle the case when occlusion causes a large region of geometry to be missing. Fig. 19 shows an example where rays go through the object because part of it is occluded and therefore cannot be reconstructed from a depth map. Some holistic single view mesh completion methods may help solve this problem, but this is beyond the scope of this paper.

One invisible lamp While our one invisible lamp assumption works well practically, it can cause errors in specific regions. One example is shown in Fig. 1 (c) in the main paper. Compared to the real photo Fig. 1 (c.1), the lamp near the bed projects a wrong shadow on the wall because there are several small lamps on the ceiling lining against the wall in the real environment, while our method only predicts the major bright invisible lamp on top of the ceiling.

Future works Currently, our framework can only handle a single image as the input. However, multi-view inputs can potentially lead to more complete and more accurate geometry reconstruction and more observation of the intensity distribution across the room. Therefore, it will be interesting to see how these multi-view inputs can help improve the indoor light editing results.

References

1. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 16
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 3
3. Li, Z., Shafiei, M., Ramamoorthi, R., Sunkavalli, K., Chandraker, M.: Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image (2020) 2, 8, 13

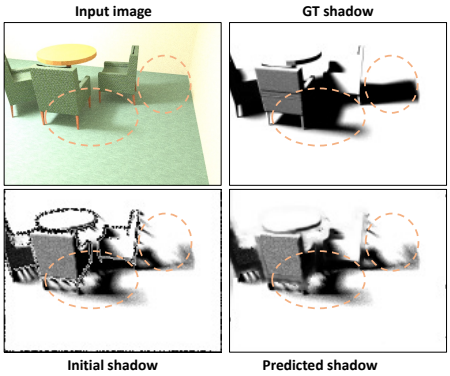

Fig. 19. Our shadow rendering framework cannot handle the situation when a large part of the object is not reconstructed due to occlusions, as shown in the errors in the orange circles.

4. Li, Z., Yu, T.W., Sang, S., Wang, S., Song, M., Liu, Y., Yeh, Y.Y., Zhu, R., Gundavarapu, N., Shi, J., Bi, S., Xu, Z., Yu, H.X., Sunkavalli, K., Hašan, M., Ramamoorthi, R., Chandraker, M.: OpenRooms: An end-to-end open framework for photorealistic indoor scene datasets. In: CVPR (2021) 1, 12, 13, 14, 16, 17
5. Niklaus, S., Mai, L., Yang, J., Liu, F.: 3d ken burns effect from a single image. ACM Transactions on Graphics (TOG) **38**(6), 1–15 (2019) 6
6. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: ICCV. pp. 12179–12188 (2021) 2, 12
7. Rother, C., Kolmogorov, V., Blake, A.: " grabcut" interactive foreground extraction using iterated graph cuts. ACM transactions on graphics (TOG) **23**(3), 309–314 (2004) 16
8. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M., Nardi, R.D., Goesele, M., Lovegrove, S., Newcombe, R.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019) 7
9. Wang, Z., Philion, J., Fidler, S., Kautz, J.: Learning indoor inverse rendering with 3D spatially-varying lighting. In: ICCV (2021) 10