A Real-time Method for Inserting Virtual Objects into Neural Radiance Fields

Keyang Ye[1], Hongzhi Wu[1], Xin Tong[2], Kun Zhou[1]

[1] State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China

[2] Microsoft Research Asia, Beijing, China

kunzhou@acm.org

Abstract

We present the first real-time method for inserting a rigid virtual object into a neural radiance field, which produces realistic lighting and shadowing effects, as well as allows interactive manipulation of the object. By exploiting the rich information about lighting and geometry in a NeRF, our method overcomes several challenges of object insertion in augmented reality. For lighting estimation, we produce accurate, robust and 3D spatially-varying incident lighting that combines the near-field lighting from NeRF and an environment lighting to account for sources not covered by the NeRF. For occlusion, we blend the rendered virtual object with the background scene using an opacity map integrated from the NeRF. For shadows, with a precomputed field of spherical signed distance field, we query the visibility term for any point around the virtual object, and cast soft, detailed shadows onto 3D surfaces. Compared with state-of-the-art techniques, our approach can insert virtual object into scenes with superior fidelity, and has a great potential to be further applied to augmented reality systems.

Keywords: Neural radiance field, all-frequency rendering, shadow, augmented reality.

1. Introduction

Neural Radiance Field (NeRF) is a popular technique for novel view synthesis, by representing a scene with implicit fields of density and view-dependent color, trained end-to-end with respect to input images [28]. Substantial research efforts have been made to extend the original work to different scenarios (e.g., dynamic scenes [32], human bodies [31] or illumination variations [27]).

Despite that NeRF is a promising, novel representation with rich information about a scene, its application to augmented reality (AR) is still limited. Related work focuses on inserting a NeRF into another one [40], or a NeRF onto a background photograph [9]. To our knowledge, very few existing techniques exploit the information in a NeRF to perform virtual object insertion, a classic AR task.

While our task looks straightforward at first glance, a number of challenges arise in inserting a virtual object into a NeRF. First, the input NeRF may not completely cover the complete lighting information of a scene. How to compensate for the potentially missing light sources? Second, it is difficult to represent and/or precompute near-field lighting in an efficient manner. Moreover, complex occlusion/shadowing effects between the virtual object and the NeRF must be modeled with high fidelity and performance.

To tackle the above challenges, we present in this paper the first real-time method for inserting a virtual object into a neural radiance field, which produces realistic rendering with shadowing and occlusion effects, as well as allows interactive manipulation of the virtual object. Our method takes as input an HDR NeRF and one or more virtual objects only, and requires a modest overhead for precomputation and storage. To account for light sources not covered by NeRF, we estimate a distant environment lighting with inverse rendering. To efficiently handle near-field lighting, we sample the NeRF and combine with the environment lighting as the incident lighting, expressed in spherical Gaussians (SG) for high-performance rendering. To rapidly model shadowing effects, we precompute the self-visibility of the virtual object, and store the result as a field of spherical signed distance fields (SSDF) [36], which can be queried efficiently in real time. Our method compares favorably with state-of-the-art techniques in generating high-fidelity insertion results.

Our main contribution is the first complete framework to insert a virtual object into a neural radiance field in real time, an increasingly popular representation. The rest of this paper is structured as follows. Sec. 2 discusses the related work, and Sec. 4 describes our method. In Sec. 5, we present experiment results for both synthetic and real scenes. Finally, Sec. 6 concludes the paper.

2. Related Work

Below we review three categories of work most related to this paper.

Lighting estimation. Here previous work can be divided into scene-based and object-based methods. Scene-

based methods estimate the lighting from partial view(s) of the scene, which is similar to the task of image completion. The missing part is approximated by directly copying from input images [21], or by searching a panorama database for an environment map similar to the input [19]. Recently, deep learning is employed to fill in the missing information. Convolutional Neural Network (CNN) or Generative Adversarial Network (GAN) are used to predict an environment map from input images with limited fields of view [15, 22, 42, 24]. Garon et al. [16] take images with coordinate masks and corresponding local patches as input to a neural network to predict 2D spatially-varying lighting expressed in spherical harmonics (SH). However, learning-based methods suffer from weak generalization ability, when the input images differ considerably from the training set. For example, the models trained on indoor scene datasets are difficult to generalize to outdoor [42, 14].

On the other hand, object-based methods estimate lighting from the appearance of object(s) presented in the scene. A common technique is to jointly optimize the object material and the environment lighting by inverse rendering [44, 34, 6, 43, 45]. This is a highly ill-posed problem, and its solution often requires strong priors, such as smoothness [44] or Lambertian only [4]. Recently, deep learning has also been applied. Much of related work is targeted for human faces, due to the existing prior knowledge about their shapes and appearance [35, 8].

Our method applies inverse rendering to estimate a distant environment lighting from a NeRF to compensate for potential missing lights not recorded in the NeRF. We take both the near-field lighting from NeRF and the distant environment lighting into consideration and build an inverse rendering pipeline, similar to [45].

Visibility estimation. Most contemporary AR systems estimate depths for visibility computation. Depths can be directly acquired with specialized devices [20, 5, 10]. To reduce the hardware requirement, single-view approaches apply deep learning [1, 17] or employ inverse rendering that adds depths as additional unknowns to a joint optimization [23]. When a sequence of images is available, Multi-View Stereo (MVS) estimates depths by matching pixels [37] or features [41]. Waston et al. [39] sidestep the depth estimation and directly predict the occlusion mask from a feature map. However, these methods still fall short in some cases, especially on translucent objects or objects with tiny structures, such as leaves on plants. In comparison, our method exploits the implicit geometry in a NeRF, which can handle cases that are challenging for traditional explicit representations.

Real-time shadows. Shadow field [46] precomputes visibility maps of samples from concentric shells surrounding the virtual object and use SH as basis to compress them. However, the approach only supports soft shadows when us-

ing low order SH basis. Wang et al. [36] introduce spherical signed distance fields (SSDF) for high-frequency shadows. Their method precomputes the SSDF at every vertex that receives shadows, and compresses them with PCA method, which requires both the scene and the object being static. Kei et al. [18] approximate objects with a set of spheres and project them to the hemisphere integral region of the shading point as occluded patches. Dynamic objects are supported at the cost of intensive computation. We propose an SSDF field, which combines the idea of shadow field and SSDF, to efficiently render shadows cast by the virtual object.

3. Preliminaries

A NeRF represents a scene with a volumetric field $\mathcal{F}$ of density σ, and color $\mathbf{c}$ which varies with a view direction $\mathbf{d}$. For a ray $\mathbf{r}(t) = \mathbf{o} + \mathbf{d}t$ emitted from camera center $\mathbf{o}$ along $\mathbf{d}$, the rendered color $\hat{\mathbf{C}}(\mathbf{d}; \mathbf{o})$ of a NeRF is computed with simple volumetric rendering [28]. The opacity $\hat{O}(\mathbf{d}; \mathbf{o})$ and depth $\hat{D}(\mathbf{d}; \mathbf{o})$ are computed in a similar fashion: $\hat{O}(\mathbf{d}; \mathbf{o}) = \sum_{k=1}^{K} T(t_k) \alpha(\sigma(t_k)\delta_k)$ and $\hat{D}(\mathbf{d}; \mathbf{o}) = \sum_{k=1}^{K} T(t_k) \alpha(\sigma(t_k)\delta_k) t_k$. Here t_k is a distance of a point sample traveled along the ray, $t_k \in [t_n, t_f]$, where t_n and t_f are the near and far distances of the intersection of the NeRF boundary and the ray. Please refer to Fig. 2 for an illustration of t_k. $T(t_k)$ is the cumulative transmittance, $\sigma(t_k)$ are the density at the sample point on the ray $\mathbf{r}(t)$. In addition, $\alpha(x) = 1 - e^{-x}$ and $\delta_k = t_{k+1} - t_k$ is the distance between adjacent two samples along the ray.

NeRF can be trained from a set of images with known intrinsic and extrinsic camera parameters. While low-dynamic-range (LDR) images are usually used, they are not suitable for physically based rendering. Therefore, we use high-dynamic-range (HDR) NeRF as input to our pipeline throughout the paper. Moreover, we employ instant-ngp [30] as the implementation, due to its excellent performance.

Assumptions. We assume that the virtual object is rigid and opaque. Also its size is relatively small with respect to the input NeRF, so that the incident lighting (ignoring occlusion) does not change over its surface. For the input NeRF, we assume that the geometry incorporated in it can be sampled as surface points, and its materials are Lambertian. In the whole pipeline, we only consider the direct illumination and ignore interreflection effects.

Note that the above assumptions may be loosened, at the expense of more runtime computations. For example, one may build the incident lighting at sampled locations across the object surface, and interpolate them to obtain the result for any point. For the surface assumption of the NeRF, one can remove it by switching to a more involved inverse rendering computation, which takes scattering into account. On the high level, our framework is not limited to the above

assumptions and can be extended to handle more diverse cases.

4. Our method

4.1. Overview

We take as input an HDR NeRF (denoted as $\mathcal{F}$) and a rigid virtual object with geometry (represented as a mesh) and appearance (parameters stored in texture maps). First, to compensate for potential lighting not covered by the NeRF, we extrapolate it to an additional environment lighting, and then combine both into an incident lighting expressed in SGs for fast evaluations (Sec. 4.2). Next, to facilitate rapid shadow computation, we precompute the visibility field around the virtual object as a field of SSDF (Sec. 4.3). At runtime, we render the virtual object with estimated incident lighting and the visibility field, and composite with the original NeRF as the output in real time (Sec. 4.4). A graphical illustration of the pipeline is shown in Fig. 1.

Note that an alternative option is to precompute an field of incident lighting, expressed in SGs. However, the non-linear nature of SGs makes the interpolation difficult, which motivates our current approach for lighting estimation.

4.2. Environment lighting estimation

Ideally, all incident lighting of the virtual object comes from $\mathcal{F}$. In practice, however, $\mathcal{F}$ may not cover the complete surroundings: there might be light sources in the scene, which are not recorded by $\mathcal{F}$. To alleviate this issue, we extrapolate $\mathcal{F}$ to an additional distant environment lighting to fully cover the virtual object. Specifically, we formulate the incident lighting at a surface point $\mathbf{x}$ of the virtual object along a direction $\mathbf{d}$ as follows:

$$L_i(\mathbf{d}; \mathbf{x}) = (1 - \hat{O}(\mathbf{d}; \mathbf{x}))L_{\text{env}}(\mathbf{d}) + \hat{O}(\mathbf{d}; \mathbf{x})L_{\text{nerf}}(\mathbf{d}; \mathbf{x}),$$
(1)

where $\hat{O}(\mathbf{d}; \mathbf{x})$ is the NeRF opacity according to Sec. 3, and L_i, L_{env}, L_{nerf} are the incident lighting, the distant environment lighting and the near-field lighting from $\mathcal{F}$, respectively.

For the environment lighting, we represent it as the sum of $M = 32$ spherical Gaussian (SG) lobes, which strike a good balance between the fidelity and rendering efficiency [36]:

$$L_{\text{env}}(\mathbf{d}) = \sum_{k=1}^{M} G(\mathbf{d}; \mathbf{p}_k, \lambda_k, \boldsymbol{\mu}_k),$$
(2)

where $\mathbf{p}_k \in \mathbb{S}^2$ is the center, $\lambda_k \in \mathbb{R}^+$ is the sharpness, $\boldsymbol{\mu}_k \in \mathbb{R}_+^n$ is the amplitude, for a particular SG.

To estimate the SG parameters of an environment lighting, we first randomly sample views that pointing towards the center of the NeRF volume, with a distance of half of the NeRF grid size. The view will be rejected if the image quality of the corresponding NeRF rendering result is not sufficient (BRISQUE [29] score > 50). We repeat the process until we have 100 sampled views. For the rendered image at each sampled view, we randomly sample 64 pixels that pass through $\mathcal{F}$. For each such pixel, we compute a surface point sample as: $\mathbf{x}_{\text{surf}} = \mathbf{o} + \hat{D}(\mathbf{d}; \mathbf{o})\mathbf{d}$, similar to [44], where $\mathbf{o}$ is the camera center. The normal of this sample is estimated by the gradient of density as $\mathbf{n}(\mathbf{x}_{\text{surf}}) = -\nabla\sigma(\mathbf{x}_{\text{surf}})$. Finally, the environment lighting is linked with the rendered pixels ($L_o(\mathbf{x})$) according to the rendering equation as follows, which allows us to perform inverse rendering to optimize the SG parameters:

$$L_o(\mathbf{x}) = \frac{\mathbf{a}(\mathbf{x}; \boldsymbol{\Theta}_a)}{\pi}(E_{\text{nerf}}(\mathbf{x}) + E_{\text{env}}(\mathbf{x}; \boldsymbol{\Theta}_{\text{env}})).$$
(3)

Here $\mathbf{a}$ is the albedo, represented as a 6-layer MLP [45] (whose parameters are $\boldsymbol{\Theta}_a$) and jointly optimized along with the environment lighting. $\boldsymbol{\Theta}_{\text{env}}$ are the SG parameters. E_{nerf} is the irradiance integrated from $\mathcal{F}$ via Monte-Carlo sampling, which is only computed once throughout the optimization. For E_{env}, we first cache the NeRF opacity at each sample point into texture maps, and then perform importance sample according to the mixture of SGs to compute the integral during the optimization, similar to the method of Zhang et al. [45].

4.3. Visibility field precomputation

Once the environment lighting is estimated, the next step is to precompute the visibility for any point around the virtual object, so that such information can be rapidly retrieved during runtime for shadow computation. Inspired by shadow field [46], we precompute a field of SSDF around the virtual object, which allows fast rendering with incident lighting expressed as SGs [36].

First, a spherical signed distance field at a point $\mathbf{x}$ around the virtual object is defined as follows. Given a ray from $\mathbf{x}$ along the direction $\mathbf{d}$, the SSDF $S(\mathbf{d}; \mathbf{x})$ is the minimal angle between $\mathbf{d}$ and the direction from $\mathbf{x}$ to the the silhouette of the virtual object. The angle is positive when $\mathbf{d}$ does not intersect with the object, and negative otherwise. Interested readers are directed to [36] for a detailed derivation.

Next, we precompute the SSDFs at sampled points around the virtual object. Similar to [46], we use the points on a uniform 3D grid of $16 \times 16 \times 16$, whose center coincides with that of the object $\mathbf{x}_o$. The length of the grid is 3 times the radius of the bounding sphere of the object. Furthermore, we perform principal component analysis (PCA) to compress all sampled SSDFs to 1.8% of the original size.

At runtime, the precomputed SSDF samples are processed to a continuous field for pixel-level rendering. For a point inside the grid, the SSDF can be obtained via a tri-linear interpolation of the SSDFs at related sampled points.
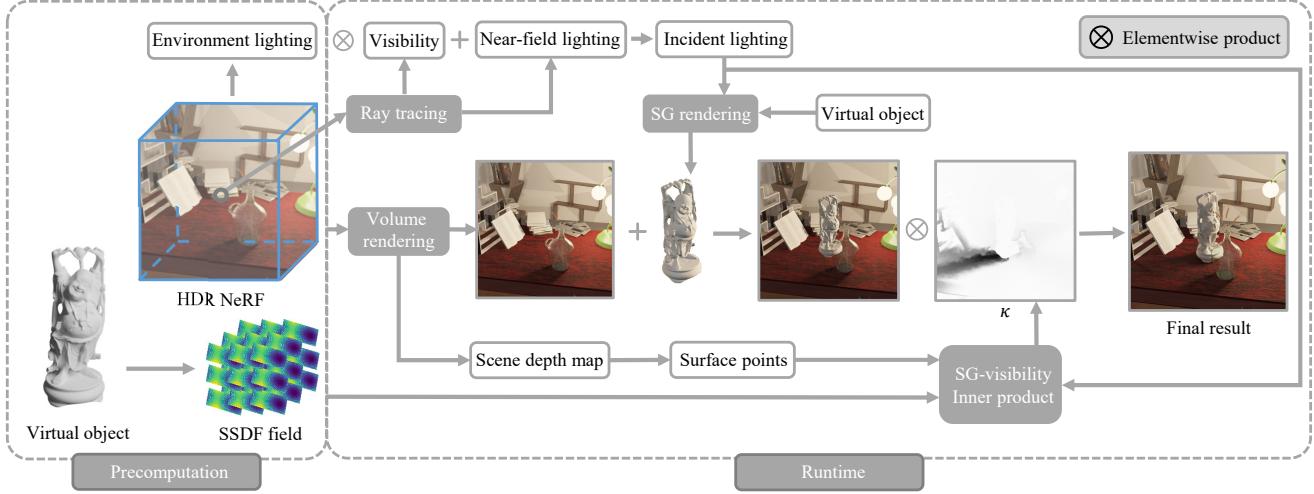
Figure 1. The full pipeline of our method. Our method takes an HDR NeRF and a rigid virtual object as input. We estimate the environment lighting to compensate for potential lighting not covered by the NeRF. Next, we precompute the visibility field of the virtual object as a SSDF field. At runtime, we perform ray tracing from the virtual object center to obtain the near-field lighting and the visibility of NeRF, and compute the incident lighting fitted with spherical Gaussians (SG) to fast render the virtual object. The rendered virtual object will be composited into the NeRF by blending. We also use the incident light and the SSDF field to compute κ for shadows at each pixel (Eq. (6)), and multiply the pixel color by the corresponding κ to produce the final result.

For a point $\mathbf{x}_f$ outside the grid, we snap to $\mathbf{x}_b$, the nearest point in the grid. We adjust the SSDF at $\mathbf{x}_b$ with simple geometric relations to approximate that of $\mathbf{x}_f$.

4.4. Real-time virtual object insertion

With the precomputed SSDF field, we perform virtual object insertion as follows. We compute the incident lighting by ray-tracing at the center of the virtual object according to Sec. 4.2 and fit it with SGs (which we refer to it as SG updating), render the object, composite it into the NeRF and add shadows at each frame. Note that for SG fitting, we adopt the estimates from the previous frame as initial values, both for temporal stability and faster convergence. Below are detailed descriptions.

Virtual object rendering. To model the object appearance, we adopt the simplified Disney BRDF model [7] with roughness r, metallic m and diffuse albedo $\mathbf{a}$, and assume a fixed $F_0 = 0.02$ in the Fresnel term. As in previous work [43, 36], the BRDF f_r and the cosine term $(\boldsymbol{\omega}_i \cdot \mathbf{n})$ can be converted to mixtures of SGs as well. Therefore, the rendering computation is approximated as a rapid inner product of SGs using a fragment shader:

$$L_o(\boldsymbol{\omega}_o; \mathbf{x}) = \sum_{k=1}^{M} (G(\boldsymbol{\omega}_i; \mathbf{p}_k, \lambda_k, \gamma_k(\mathbf{x})\boldsymbol{\mu}_k) \otimes f_r) \cdot (\boldsymbol{\omega}_i \cdot \mathbf{n}),$$

(4)

where $\mathbf{p}_k, \lambda_k$ and $\boldsymbol{\mu}_k$ are the SG parameters of the incident lighting at the virtual object center. γ_k attenuates the final result to account for self-shadows, which we will detail in the description of shadow computation. $\otimes$ represents
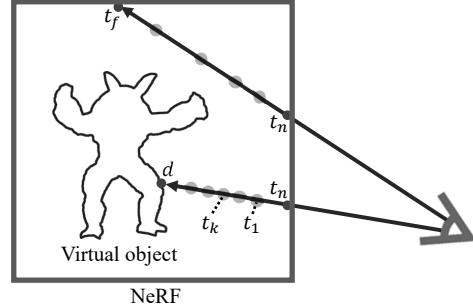

Figure 2. The illustration of sampling points for volume rendering. Rays are emitted from the camera center to the NeRF to render the view. For rays not intersecting with the virtual object, we sample points with the distance range $t_k \in [t_n, t_f]$, where t_n and t_f are the near and far distances of the intersection of the NeRF boundary and the ray. Otherwise, we sample points with the distance range $t_k \in [\min(d, t_n), \min(d, t_f)]$, where d is the depth value of the virtual object, to eliminate the part of the NeRF occluded by the virtual object.

direction-wise multiplication (SG product). As the BRDF is represented with SGs, this allows direct editing of appearance in real time.

Compositing. Here we mix the rendering results of the input NeRF and the virtual object. First, for a pixel whose camera ray hits the virtual object, we adjust the integration range t to $[\min(d, t_n), \min(d, t_f)]$, where d is the depth of the object at the current pixel. The idea is to eliminate the part of the NeRF occluded by the virtual object. Please refer to Fig. 2 for an illustration. We then perform conventional NeRF rendering to obtain a color and an opacity map, $\mathcal{I}_c$

and $\mathcal{I}_\alpha$. Next, we blend in the rendering result of the virtual object (without shadows) $\mathcal{I}_v$ as:

$$\mathcal{I}_\alpha\mathcal{I}_c + (1 - \mathcal{I}_\alpha)\mathcal{I}_v. \quad (5)$$

Shadow computation. We handle two types of shadows in our pipeline: the shadow from the virtual object to the NeRF, and the self-shadow of the virtual object.

For the first type of shadows, similar to previous work [38], we compute an attenuation factor κ, as the ratio of the irradiance value before and after the virtual object insertion:

$$\kappa(\mathbf{x}) = \frac{\int_{\Omega^+} L_i(\boldsymbol{\omega}_i; \mathbf{x}_o) V(\boldsymbol{\omega}_i; \mathbf{x}) (\boldsymbol{\omega}_i \cdot \mathbf{n}) \, d\boldsymbol{\omega}_i}{\int_{\Omega^+} L_i(\boldsymbol{\omega}_i; \mathbf{x}_o) (\boldsymbol{\omega}_i \cdot \mathbf{n}) \, d\boldsymbol{\omega}_i}. \quad (6)$$

Here $\mathbf{x}_o$ is the object center. V is the binary visibility term: it is 0 if occluded by the virtual object, and 1 otherwise. For each pixel in $\mathcal{I}_c$, we compute its corresponding 3D location $\mathbf{x}$ from its depth using the same method in Sec. 4.2, and then attenuate its rendered result by a corresponding κ.

Eq. (6) can be rapidly computed thanks to the SG basis. For its numerator, we first multiply L_i with the cosine term as a product of SGs, and then compute as follows:

$$\int_{\Omega} (\sum_{k=1}^{M} G(\boldsymbol{\omega}_i; \mathbf{p}_k, \lambda_k, \boldsymbol{\mu}_k)) V(\boldsymbol{\omega}_i) d\boldsymbol{\omega}_i$$
$$\approx \sum_{k=1}^{M} \boldsymbol{\mu}_k f_h(S(\mathbf{p}_k), \lambda_k), \quad (7)$$

where $f_h(\theta, \lambda)$ is a precomputed table [36] related to the spherical signed distance (SSD) θ and the sharpness λ of an SG. For the denominator, it can be quickly evaluated in a manner similar to Eq. (4).

For adding the self-shadows, we first attenuate each SG for the incident lighting with a factor γ_k, and then render the virtual object. γ_k is the ratio of the integral of the unobstructed region to that of the complete upper hemisphere:

$$\gamma_k(\mathbf{x}) = \frac{\boldsymbol{\mu}_k f_h(S(\mathbf{p}_k; \mathbf{x}), \lambda)}{\boldsymbol{\mu}_k f_h(\frac{\pi}{2}, \lambda)} = \frac{f_h(S(\mathbf{p}_k; \mathbf{x}), \lambda)}{f_h(\frac{\pi}{2}, \lambda)}. \quad (8)$$

Note that here we do not attenuate with κ to avoid computing the shadows twice.

5. Results

All experiments are conducted on a workstation with an Xeon(R) Platinum 8352V CPU, 90GB memory and an NVIDIA RTX 4090 GPU. Our pipeline achieves a performance of 18 frames per second on average at a resolution of 1280×720. Specifically, it takes about 40ms for instant-ngp to render the NeRF. And it takes 15ms for virtual object insertion, including 4ms for incident lighting computation,

7ms for SG updating (except that 84ms is needed for the first frame), and 4ms for the rendering, compositing and shadow casting of the virtual object. The average GPU usage is 55%, which shows room for future improvement over memory access.

For precomputation, it takes about 25 minutes to estimate the environment lighting from an input NeRF. For a virtual object, we spend 8 minutes to precompute the SSDF field, which takes up 9MB. Note that the SSDF field can be used with any NeRF.

In the following, we describe the data related to our experiments, including synthetic and captured HDR NeRFs, in Sec. 5.1. Ablation studies are demonstrated in Sec. 5.2. Finally, we compare against state-of-the-art methods in Sec. 5.3.

5.1. Data

We build four synthetic HDR NeRFs by training using images rendered from multiple views with Blender Cycles [11]. For PLANTS, all light sources can be observed in the rendered images, and therefore the environment lighting should be totally black. For the other 3 NeRFs (BOOKS, DOG and CUPS), there are some light sources that do not appear in any of the rendered images, which are considered as the environment lighting.

In addition to synthetic NeRFs, we also construct two NeRFs by training with captured HDR photographs from two real scenes (HORSE and SHEEP). We set the camera to the bracketing mode with the step of 1.5 EV and shoot 9 images with different exposures for each camera pose. Each HDR image is recovered from 9 LDR images using [12].

To provide the ground-truth for object insertion experiments, we 3D-print several models and uniform sprayed them with paints to simulate different materials. We apply Ma et al. [25] to measure the material parameters, including roughness and metallic, and use the same parameters for virtual objects rendering. The 3D printed models are placed in the real scene, and images are taken around them as the ground-truth. We use the images without the presence of the model to train NeRFs, and compare object insertion results with the images with the physical model in place. The sizes of virtual objects are adjusted manually to ensure that it is closed to the ground-truth.

5.2. Ablation studies

We first perform ablation experiments to show the intermediate results of each step described in Sec. 4.4. In Fig. 3, the virtual object shows all-frequency effects with realistic reflections. We can see the virtual object through the glass bottle in the BOOKS scene, which is a challenging task for traditional AR methods [26]. The shadows computed by our method also considerably enhanced the visual realism.

Next, we show the impact of our estimated environment

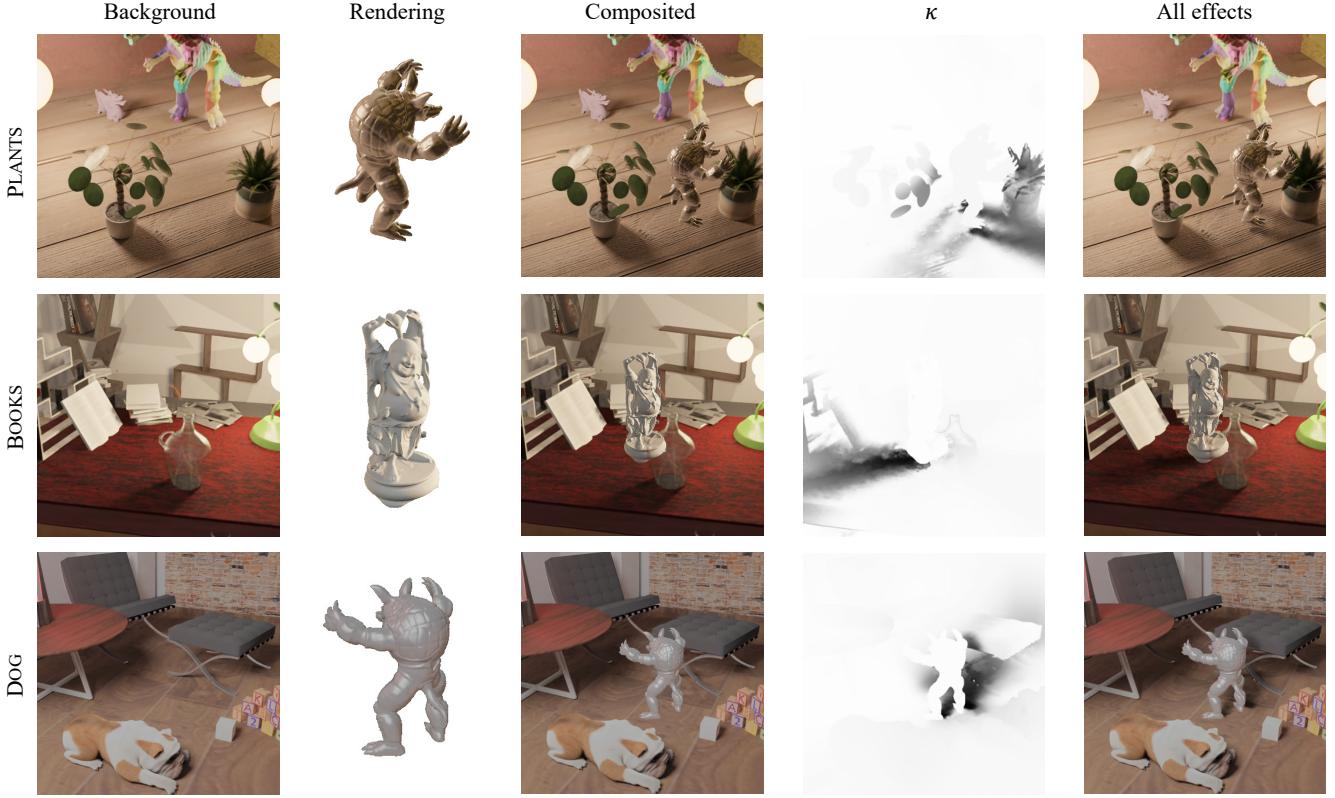| Background	Rendering	Composited	κ	All effects

Figure 3. Visualization of intermediate and final object insertion results of the PLANTS, BOOKS and DOG synthetic scenes. From left column to right, background images, virtual objects rendering results, compositing results with occlusion or translucency effects, κ (Eq. (6)), object insertion results with all effects.

lighting as well as the near-field lighting represented in the NeRF in Fig. 4. Without the environment lighting which estimates the sources not covered in the NeRF, the virtual object appears darker in the BOOKS and DOG scene. Moreover, we compare our method with PhySG [43] and InvRender [45] on environment lighting estimation. These three methods have a similar estimation framework, and all represent the environment lighting as 32 SGs in our experiments. PhySG does not take into account the near-field lighting and visibility. InvRender roughly models the near-field lighting by training a MLP to cache ray-tracing results. Our method more accurately model the near-field lighting (especially the visibility, cached to texture maps), and sparsely sample the surface points to reduce memory consumption. In the PLANTS scene, all light sources are included in the NeRF, and therefore the ground-truth environment lighting is totally black. PhySG treats the light sources in NeRF as the environment lighting. The virtual object is brighter than the ground-truth in the PLANTS scene, while the self-shadow is too dark in the BOOKS scene. For InvRender, caching near-filed lighting with MLP by tracing 16 rays at each point is too rough for complex scene-level NeRFs, which leaves a lot of ambiguity during optimization. Our method provides the best results, and the estimation is not affected by the

light sources incorporated in the NeRF.

In addition, we evaluate the impact of the number of pixels and rays over the estimated environment lighting in Fig. 5. If we sample less pixels, the estimation will suffer from the ambiguity of albedo and shading. If we trace less rays at each point, the near-field lighting is less accurate, resulting in a highly noisy estimate. In comparison, sampling fewer pixels has a greater impact than sampling fewer rays. Sampling more pixels and rays usually leads to slightly better results, but bears the increasing costs of time and memory.

To validate the our lighting model in Eq. (1), we show the near-field lighting (from NeRF), estimated environment lighting, full incident lighting (before and after SG fitting), and the ground-truth in Fig. 6. Our method generates high-quality incident lighting close to the ground-truth. After fitting with SGs, the main light sources are preserved. We notice that there are some extra floaters (PLANTS scene) and missing parts (BOOKS scene) in the near-field lighting. Nevertheless, they do not notably influence the rendering as the brightness is relatively low. Also, the estimated environment lighting well compensates for the lighting not covered by NeRF.
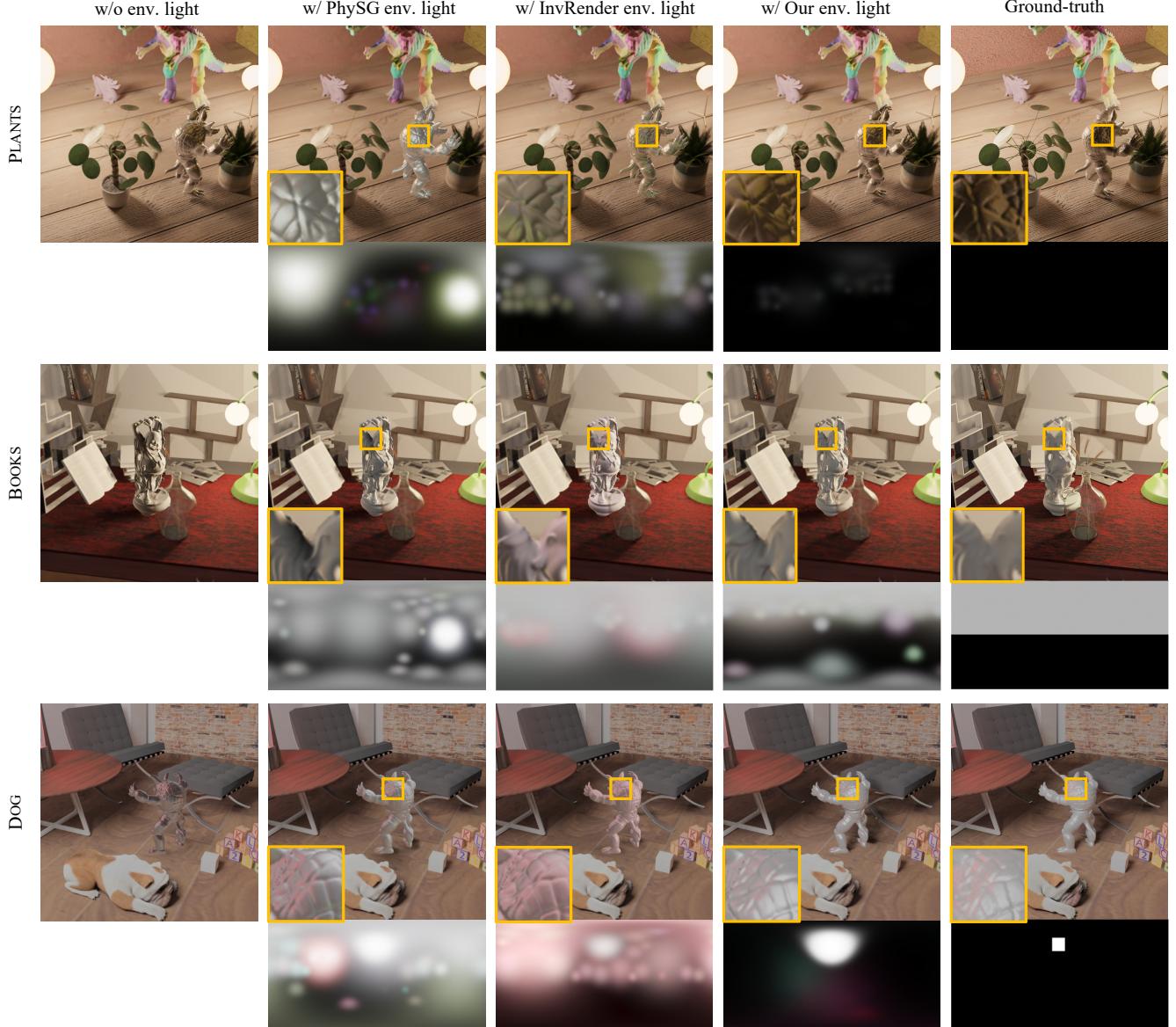
To validate our method for extrapolating SSDF

Figure 4. A qualitative comparison of using different environment lighting in PLANTS, BOOKS and DOG synthetic scenes. In the first column, virtual objects are rendered by only using the NeRF near-field lighting. From the second left column to the fourth column, we compensate different estimated environment lighting estimated by the method of PhySG [43], InvRender [45] and our proposed method. The last column is the ground-truth rendered by Blender Cycles. Some details are magnified for better comparison. The environment lighting is shown by equirectangular projection under the rendered images.

Figure 5. Estimating environment lighting by different sampling strategies. All the results are estimated in DOG. The result of our proposed method is shown on the second from left, which samples 64 pixels per view and traces 32768 rays at each surface point corresponding to the pixel. Results estimated by other sampling strategies are shown on the right side of the dashed line.

Figure 6. Visualization of the lighting at the virtual object center. From left column to right, near-field lighting, environment lighting, incident lighting, SG fitted incident lighting and the ground-truth lighting rendered by Blender Cycles. The incident lighting is the blend of the near-field and environment lighting according to the NeRF opacity.

Figure 7. Comparison of shadows generated by using the $16 \times 16 \times 16$ SSDF field that obtains SSD values at far positions by clamping to the boundary values (a), truncating to positive infinity (b) and approximating from the boundary values (c), and shadows generated by using the $64 \times 64 \times 64$ SSDF field that interpolates SSDF at any position (d). The ground-truth is rendered by Blender Cycles.

(Sec. 4.3), we compare in Fig. 7 the shadows generated by our parameters and using a $64 \times 64 \times 64$ SSDF field with an extended range of $-6r_{\mathrm{obj}}$ to $6r_{\mathrm{obj}}$, where r_{obj} is the radius of the virtual object bounding sphere, so that the SSDF is interpolated at any pixel in the rendered image. There are no significant differences between the results computed with two sets of parameters, while the simple clamping method generates unsatisfactory results in regions far away from the virtual object. Compared to the ground-truth rendering results, our shadows appear slightly darker, due to the approximation of the inner product of the environment lighting and the visibility term.

5.3. Comparisons

We first compare our approach with two state-of-the-art methods, Li et al. [23] and Zhan et al. [42]. Both take RGB images as input and employ neural networks to estimate the lighting. The former outputs 2D spatially-varying incident lighting, from a joint optimization of material parameters and depths. And the latter only outputs one incident lighting. For comparison, we first render (for synthetic scenes) or capture (for real scenes) images of a scene without the virtual object, and use them as input for both methods. The estimated lighting are represented as SGs, so that we can render the virtual object as described in Sec. 4.4 to compare across different methods. Depth test is performed in [23] to account for occlusions. Since the method of [42] does not provide any geometric information, we directly place the virtual object on top of the background. We do not compare our shadows with the baseline methods, as both of them generate shadows on manually specified planes. Instead, we compare our shadows with the ground-truth.
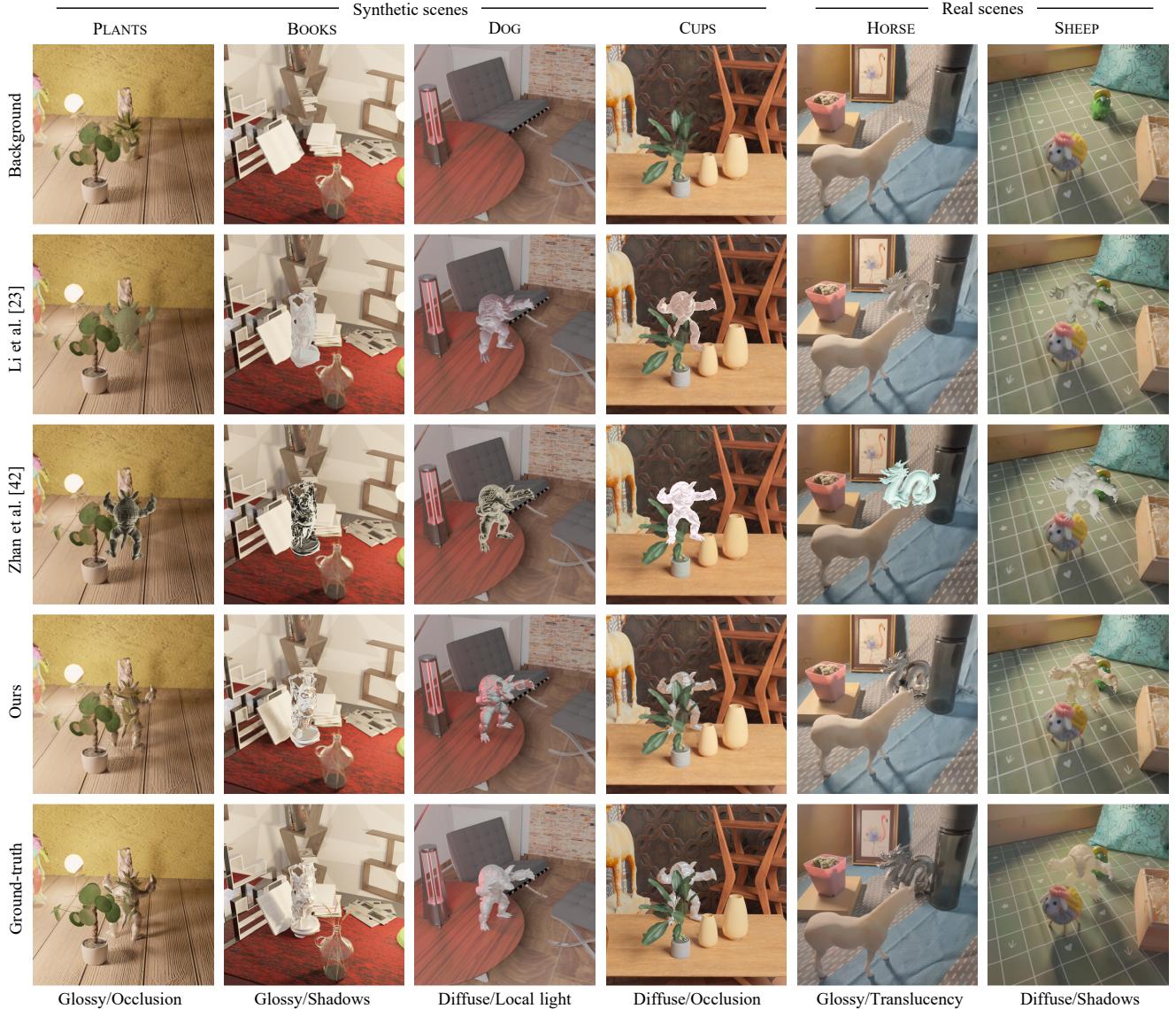
Figure 8. Comparison with the work of Li et el. [23] and Zhan et al. [42] in both synthetic scenes and real scenes. The materials of virtual objects and the noteworthy effects are indicated under each column.

Fig. 8 shows the comparisons on synthetic and real scenes. Virtual objects rendered with [23] or [42] can be easily distinguished from the background. These two methods are trained with indoor image datasets and are likely to estimate results with considerable bias. In comparison, our method is able to robustly estimate the incident lighting. Our rendering results exhibit realistic reflections and shadows that closely resemble the ground-truth, regardless of the material, e.g. diffuse in DOG or glossy in BOOKS. Even for scenes with complex lighting, such as multiple light sources in PLANTS and the environment lighting with local lights in DOG, our method is able to produce reliable results. The depth map computed by Li et al. is rough, leading to noticeable cracks or overlaps, e.g., in PLANTS and CUPS. Note

that our method supports the transparency and translucency represented in the input NeRF (see the plastic cup in HORSE in Fig. 8 and the glass vase in BOOKS in Fig. 3).

In Fig. 9, we further compare with the concurrent work of DMRF [33]. There are 3 major differences. First, based on a path-tracing framework, their performance cannot reach real time. Second, their approach does not estimate light sources not covered the input NeRF. Finally, the visibility in the NeRF is ignored, leading to artifacts like shadow leakage, as shown in Fig. 9.

Moreover, we qualitatively compare against popular AR frameworks from the industry, including Google Depth Lab [13], AR Toolbox [3] and Unity AR Foundation [2], due to the difficulty for precisely controlled experiments.
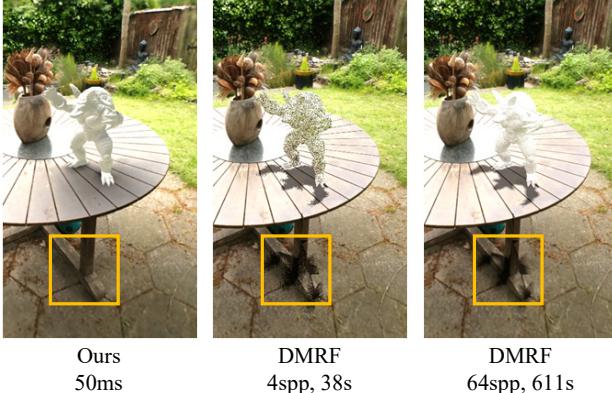
Figure 9. Comparison with DMRF [33]. It takes 50ms to render the frame by our methods while DMRF takes 38s to render a 4spp frame and 611s to render a 64spp frame. Leaked shadows are observed with their approach.
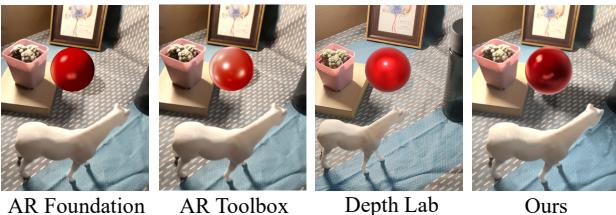

Figure 10. Comparison with popular AR frameworks in industry, including Unity AR Foundation [2], AR Toolbox [3] and Google Depth Lab [13].

As the common types of virtual object supported by all frameworks are limited, we only test a red ball in the HORSE scene. As shown in Fig. 10, our method produces more accurate and detailed rendering with soft shadows, while the counterparts only implement hard shadows on estimated planes.

5.4. Limitations

First, as shown in Fig. 11(a), certain local lighting effects may not be modeled, due to our assumption that all surface points of the virtual object shares the same incident lighting. Second, due to the inaccurate depth estimation in NeRF, there is a small amount of noise in our shadows in Fig. 11(b). Finally, we consider direct illumination only, which may lead to artifacts. For example, in Fig. 11(c), the virtual object rendered by our method shows red reflections from the wood table. However, this is incorrect, as this part of the wood table is actually occluded by shadows of the object.

6. Conclusion

We propose a real-time method for inserting virtual objects into a NeRF. Our key contribution is the first framework that fully exploits the lighting and geometry infor-
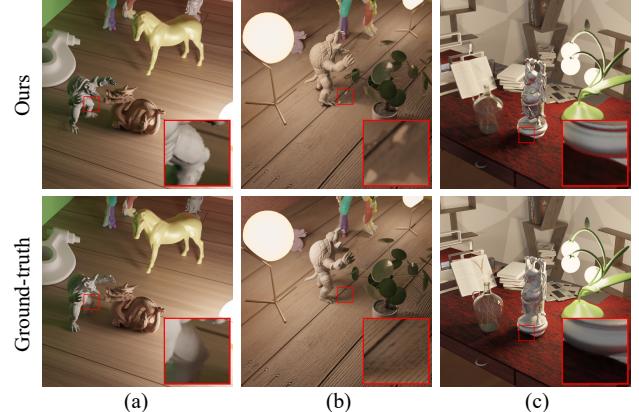

Figure 11. Failure cases: (a) shadow missing due to the assumption that all surface points of the virtual object shares the same incident lighting; (b) noisy shadow caused by inaccurate depth estimation; (c) incorrect shading due to only considering the direct illumination.

mation in NeRFs to enable realistic rendering with occlusion and shadowing effects. We outperform state-of-the-art techniques in terms of quality. Compared with existing work, combining NeRF with AR has considerable benefits and may be widely deployed with the popularity of NeRF. We hope that this paper could inspire further research along this direction.

References

[1] I. Alhashim and P. Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018. 2

[2] Unity AR Foundation. Website, 2023. https://unity.com/unity/features/arfoundation. 9, 10

[3] AR Toolbox. Website, 2023. https://play.google.com/store/apps/details?id=fr.smarquis.ar_toolbox. 9, 10

[4] D. Azinovic, T.-M. Li, A. Kaplanyan, and M. Nießner. Inverse path tracing for joint material and lighting estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2447–2456, 2019. 2

[5] S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 2

[6] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021. 2

[7] B. Burley and W. D. A. Studios. Physically-based shading at disney. In *Acm Siggraph*, volume 2012, pages 1–7. vol. 2012, 2012. 4

[8] D. A. Calian, J.-F. Lalonde, P. Gotardo, T. Simon, I. Matthews, and K. Mitchell. From faces to outdoor light

probes. In *Computer Graphics Forum*, volume 37, pages 51–61. Wiley Online Library, 2018. 2

[9] J. Cao, H. Wang, P. Chemerys, V. Shakhrai, J. Hu, Y. Fu, D. Makoviichuk, S. Tulyakov, and J. Ren. Real-time neural light field on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8328–8337, 2023. 1

[10] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5418, 2018. 2

[11] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[12] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, page 369–378. ACM Press/Addison-Wesley Publishing Co., 1997. 5

[13] R. Du, E. Turner, M. Dzitsiuk, L. Prasso, I. Duarte, J. Dourgarian, J. Afonso, J. Pascoal, J. Gladstone, N. Cruces, et al. Depthlab: Real-time 3d interaction with depth maps for mobile augmented reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 829–843, 2020. 9, 10

[14] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagné, and J.-F. Lalonde. Deep parametric indoor lighting estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7175–7183, 2019. 2

[15] M.-A. Gardner, K. Sunkavalli, E. Yumer, X. Shen, E. Gambaretto, C. Gagné, and J.-F. Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017. 2

[16] M. Garon, K. Sunkavalli, S. Hadap, N. Carr, and J.-F. Lalonde. Fast spatially-varying indoor lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6908–6917, 2019. 2

[17] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 2

[18] K. Iwasaki, W. Furuya, Y. Dobashi, and T. Nishita. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. In *Computer Graphics Forum*, volume 31, pages 727–734. Wiley Online Library, 2012. 2

[19] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics (TOG)*, 33(3):1–15, 2014. 2

[20] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE international conference on computer vision*, pages 66–75, 2017. 2

[21] E. A. Khan, E. Reinhard, R. W. Fleming, and H. H. Bülthoff. Image-based material editing. *ACM Transactions on Graphics (TOG)*, 25(3):654–663, 2006. 2

[22] C. LeGendre, W.-C. Ma, G. Fyffe, J. Flynn, L. Charbonnel, J. Busch, and P. Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5918–5928, 2019. 2

[23] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2475–2484, 2020. 2, 8, 9

[24] S. Ma, Q. Shen, Q. Hou, Z. Ren, and K. Zhou. Neural compositing for real-time augmented reality rendering in low-frequency lighting environments. *Science China Information Sciences*, 64:1–15, 2021. 2

[25] X. Ma, K. Kang, R. Zhu, H. Wu, and K. Zhou. Free-form scanning of non-planar appearance with neural trace photography. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. 5

[26] M. C. d. F. Macedo and A. L. Apolinario. Occlusion handling in augmented reality: Past, present and future. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 5

[27] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 1

[28] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[29] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012. 3

[30] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2

[31] S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021. 1

[32] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1

[33] Y.-L. Qiao, A. Gao, Y. Xu, Y. Feng, J.-B. Huang, and M. C. Lin. Dynamic mesh-aware radiance fields. *arXiv preprint arXiv:2309.04581*, 2023. 9, 10

[34] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021. 2

[35] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt. Mofa: Model-based deep con-

volutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1274–1283, 2017. 2

[36] J. Wang, P. Ren, M. Gong, J. Snyder, and B. Guo. All-frequency rendering of dynamic, spatially-varying reflectance. In *ACM SIGGRAPH Asia 2009 Papers*, New York, NY, USA, 2009. Association for Computing Machinery. 1, 2, 3, 4, 5

[37] K. Wang and S. Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *2018 International conference on 3d vision (3DV)*, pages 248–257. IEEE, 2018. 2

[38] Z. Wang, W. Chen, D. Acuna, J. Kautz, and S. Fidler. Neural light field estimation for street scenes with differentiable virtual object insertion. In *European Conference on Computer Vision*, pages 380–397. Springer, 2022. 5

[39] J. Watson, M. Sayed, Z. Qureshi, G. J. Brostow, S. Vicente, O. Mac Aodha, and M. Firman. Virtual occlusions through implicit depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9053–9064, 2023. 2

[40] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 1

[41] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 2

[42] F. Zhan, C. Zhang, Y. Yu, Y. Chang, S. Lu, F. Ma, and X. Xie. Emlight: Lighting estimation via spherical distribution approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3287–3295, 2021. 2, 8, 9

[43] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. 2, 4, 6, 7

[44] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 2, 3

[45] Y. Zhang, J. Sun, X. He, H. Fu, R. Jia, and X. Zhou. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18643–18652, 2022. 2, 3, 6, 7

[46] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum. Precomputed shadow fields for dynamic scenes. In *ACM SIGGRAPH 2005 Papers*, page 1196–1201, New York, NY, USA, 2005. Association for Computing Machinery. 2, 3