IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 September 2022, accepted 27 September 2022, date of publication 13 October 2022, date of current version 20 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3214516

RESEARCH ARTICLE

Online Illumination Learning for Interactive Global Illumination in Augmented Reality

WONJUN LEE, PILJOONG JEONG, HAJIN CHOI, JINWOO KIM, AND BOCHANG MOON, (Member, IEEE)
School of Integrated Technology, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea

Corresponding author: Bochang Moon (bmoon@gist.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant 2020R1A2C4002425; and in part by the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency under Project R2021080001.

ABSTRACT Augmenting virtual objects into a real scene requires estimating the scene illumination so that the augmented objects can become visually coherent with real objects. We propose an online technique that learns the illumination from image sequences captured by a hand-held device. We approximate the illumination with multiple linear models, and the coefficients and bandwidth parameters of the models are updated progressively in a data-driven way. Our online learning enables us to seamlessly integrate virtual objects into a real scene by rendering the objects with the estimated lights. We demonstrate that our framework can provide a high-quality global illumination result in augmented reality at interactive rates.

INDEX TERMS Augmented reality rendering, interactive augmented reality, illumination learning.

I. INTRODUCTION

Photorealistic rendering is a necessity to give an immersive experience to users. In augmented reality (AR), such photorealism can be obtained by augmenting virtual objects into a real scene while maintaining consistent lighting for real and virtual objects. However, it requires solving the technical challenge of rendering virtual objects while simulating light interaction between the virtual and real objects. Global illumination algorithms (e.g., path tracing [16]), which can accurately simulate such light interaction, have been well-established in the graphics field. However, a necessary input to these light transport algorithms, i.e., the illumination of a real scene, is typically unknown, and thus it should be estimated for a realistic rendering in AR.

A conventional approach for estimating the scene illumination is to capture incoming light at the place where a virtual object places, using a hardware device (e.g., light probes) [10]. This approach has been widely adopted in offline applications (e.g., movies) as it allows for accurate capturing of incident radiance at a specific point. However, determining the exact locations of virtual objects in advance can be infeasible when objects move, which is typical in interactive scenarios.

A lightweight approach for interactive AR applications is to estimate an environment map from a single image, representing the incident radiance at the locations where virtual objects are placed [21]. For example, one can render a virtual object efficiently using direct lighting with the map. However, its rendering quality can be degraded since a global illumination method requires complete light information that varies spatially.

Simultaneous localization and mapping (SLAM) based methods [26], [30], [37] provided an effective means to capture spatially varying illumination. Specifically, these methods often assumed that surfaces in a real scene have Lambertian reflectance and then estimated outgoing radiance on surfaces from color and depth image sequences captured by an RGBD camera. The SLAM-based techniques allow a global illumination to estimate radiance at an arbitrary point (not a specific point) on real objects thanks to the captured spatially varying illumination. Nevertheless, estimating the scene illumination accurately and producing an interactive AR rendering result remain technical challenges.

This paper proposes a SLAM-based technique that estimates the spatially-varying scene illumination more

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

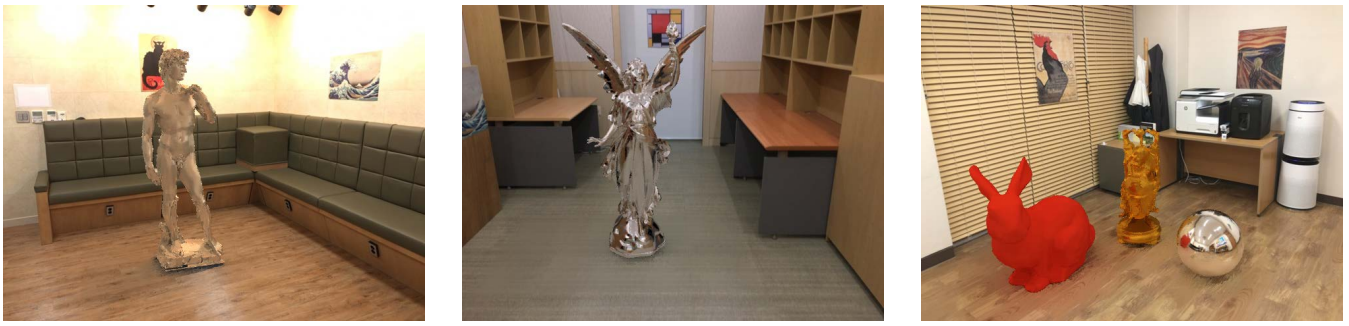


FIGURE 1. Our technique produces interactive AR rendering results using a global illumination technique that simulates light interreflections between real and virtual objects using the scene illumination estimated by our online learning. The example results are generated using a small number of samples per pixel (spp). We use the Disney BSDF [4] for glossy materials of David and Buddha models (in the left and right figures), a mirror material for Lucy and Sphere models (in the middle and right figures), and a diffuse material for the Bunny (in the right figure).

accurately from RGBD image sequences. As our main technical contribution, we present an online algorithm that learns the scene illumination progressively from a streaming input (RGBD images). Our illumination learning allows augmenting virtual objects seamlessly into a real scene, even when the objects are highly glossy or specular, thanks to our high-quality estimation on the scene illumination, as shown in Fig. 1. Our main contributions are summarized as follows:

- We approximate the scene illumination with multiple linear models, each of which estimates outgoing radiance in a local scene area. The linear models, which approximate the radiance, are created and updated progressively in a data-driven manner, and we perform this online learning at an interactive rate.
- We design an interactive global illumination that effectively renders augmented virtual objects using estimated local models.

We demonstrate that our online learning approximates the scene illumination more accurately and produces higher AR rendering quality than the state-of-the-art methods [22], [30]. Also, our online method provides an interactive visualization of our estimated scene illumination to a user as live feedback and allows a user to efficiently rescan only modified areas when the scene illumination is locally changed.

II. RELATED WORK

Achieving lighting consistency between real and virtual objects in an AR rendering scenario requires estimating the scene illumination (Sec. II-A) and conducting a global illumination that considers the light interaction between real and virtual objects (Sec. II-B). We refer to a recent survey [19] for a comprehensive overview.

A. CAPTURING SCENE ILLUMINATION

We categorize the previous methods, which capture the scene illumination, into three groups: 1) measured illumination using light probes, 2) estimated illumination, and 3) measured illumination using an RGBD camera.

1) MEASURED ILLUMINATION USING LIGHT PROBES

This approach measures incoming radiance at specific points using a specialized device (e.g., light probes) and stores the measured light into images, e.g., environment maps. A seminal work [10] demonstrated that high-quality AR rendering results could be produced due to accurate capturing of incident radiance at the positions where light probes are placed. It has also been studied to exploit multiple environment maps so that incoming radiance at a query point can be interpolated using the maps [7], [34], [35], [36]. Furthermore, Alhakamy and Tuceryan [3] estimated the direction of direct illumination from a panoramic video and approximated indirect illumination using a cube map. However, these approaches can be limited to offline scenarios where the locations of augmented virtual objects are at least approximately predetermined.

2) ESTIMATED ILLUMINATION

Estimation approaches predict incoming radiance at the points of interest from observed images even without specialized devices. Notable examples are deep learning-based approaches [14], [21], which infer an HDR environment map from a low dynamic range (LDR) image. The estimated environment map represents incoming radiance at a specific point, and one can render virtual objects using direct lighting with the environment map. These approaches are simple, but obtaining high-quality rendering results can be challenging since a single map cannot appropriately represent spatially varying illumination. Recent learning-based approaches [25], [13], [15], [22] demonstrated that a neural network could estimate spatially varying illumination (not only a single map) from a single RGB image. These learning-based techniques demonstrated impressive results even from a single image, but accurately recovering a complete illumination of a real scene is still challenging since

it should predict unobserved illumination using only limited information (e.g., a single image).

3) MEASURED ILLUMINATION USING AN RGBD SENSOR

An alternative to the aforementioned techniques is directly capturing spatially varying illumination using a sequence of images (not a single image) from an RGBD sensor. This approach takes advantage of SLAM algorithms (e.g., [20], [27]) that provides an effective means to capture illumination samples (i.e., globally registered point clouds) as well as geometries of a real scene from an RGBD image sequence. Existing approaches [26], [30], [37], which rely on SLAM methods, mainly focused on building a data structure that compactly represents the scene illumination from observed point clouds. For example, Meilland et al. [26] constructed an environment map from a set of illumination samples at a given point and exploited the map for AR rendering. Zhang et al. [37] represented the scene illumination with HDR textures by reprojecting observed illumination samples onto the estimated surfaces of a real scene. Rohmer et al. [30] proposed a sophisticated process of estimating surface normals and HDR colors of samples in a globally registered point cloud so that these enriched samples can be transformed into light representations (e.g., an environment map or voxel volume).

Analogously to the prior methods, our method relies on a SLAM algorithm that estimates a camera location and geometries of a real scene. However, unlike the previous techniques, we do not record all point samples for learning the scene illumination. Our technique approximates the scene illumination with a set of linear models and refines the models progressively and adaptively while minimizing our estimation errors via an online optimization. Our online process can also provide an interactive visualization of the estimated illumination to a user who scans an indoor scene.

B. RENDERING ALGORITHMS FOR AR

Integrating virtual objects into a real environment requires a rendering framework that considers light interaction between virtual and real objects using estimated scene illumination. Differential rendering [10] is a widely adopted framework that accounts for the illumination changes introduced by augmenting virtual objects. Given the framework, designing a global illumination technique, which efficiently computes the illumination changes, has been a technical problem, especially for an interactive rendering. For example, Knecht et al. [18] constructed two sets of virtual point lights (VPLs), each of which represents the illumination of a real scene with and without virtual objects, respectively. Then, the difference between the rendering results from each set is added to the observed image. In [11] and [12], the scene illumination was discretized using a voxel-based data structure, and the illumination change for a differential rendering was computed interactively using a real-time rendering method, voxel cone tracing (VCT) [8]. Recently, Rohmer et al. [30] demonstrated that VCT based AR rendering could be performed using a voxel-based representation where each voxel contains an averaged value of illumination samples captured by a mobile device.

Our method also provides an interactive rendering result, but unlike the existing methods, we demonstrate that our light representation allows exploiting a general global illumination technique (i.e., path tracing [16]) that can accurately simulate various lighting effects.

TABLE 1. Notations used throughout the paper.

Notation	Description
Ω^R	Set of all surface points of a real scene
Ω^V	Set of all points on virtual objects
Ω^{R+V}	$\Omega^R \cup \Omega^V$
$L_i^R(\mathbf{x}, \omega)$	Incident radiance at $\mathbf{x} \in \Omega^R$ with direction ω
$L_o^R(\mathbf{x}, \omega)$	Outgoint radiance at $\mathbf{x} \in \Omega^R$ with direction ω

III. OVERVIEW OF OUR FRAMEWORK

We target a typical AR scenario where a user navigates an indoor space (e.g., a room) with a hand-held camera. We take a sequence of color images and depths (i.e., RGBD images) as input. Fig. 2 illustrates an overview of our framework. Given the streaming input from a mobile device, we propose a new online learning technique that creates and updates linear models, each of which approximates the scene illumination in a local area (Sec. IV). Once the light learning phase completes, we start a rendering phase where virtual objects are seamlessly augmented into a real environment (Sec. V). Table 1 summarizes our notations used throughout the paper.

A. ASSUMPTIONS

Given the AR scenario, we assume the surfaces (e.g., triangular meshes) of a real scene are static, and thus we pre-compute the geometries and provide those to our learning as input. Thus, we suppose the surfaces of the scene have Lambertian reflectance, i.e., $L_o^R(\mathbf{x}, \omega) = L_o^R(\mathbf{x})$. For the sake of brevity, we will omit the direction ω when we refer the outgoing radiance $L_o^R(\mathbf{x})$.

B. MOTIVATION FOR ONLINE LEARNING

One may think that an offline process cannot be much different from online learning in practice, as we assume that the geometries of a real scene are static. The illumination, however, can change over time, even when real objects do not move. Fig. 3 illustrates an example scenario where our method adapts the estimated scene illumination for modified local areas (e.g., the poster on the right side in Fig. 3 (c)). This example indicates that, unlike offline learning, one can rescan only a local region where illumination is modified if the given space is scanned before.

IV. ONLINE ILLUMINATION LEARNING

We propose an online method that incrementally estimates the scene illumination $L_o^R(\mathbf{x})$ from an RGBD image sequence to

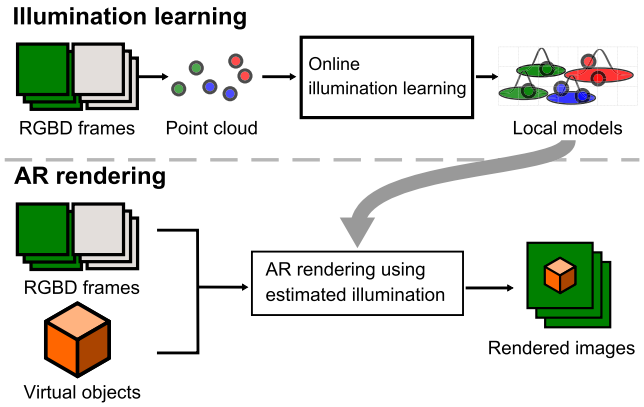


FIGURE 2. Our framework consists of two main stages: illumination learning (top) and AR rendering (bottom). We take observed illumination samples as input on the learning stage and then approximate the illumination of a real scene with multiple local models whose parameters are progressively updated. Once the illumination learning completes, we provide realistic AR rendering results to users using the estimated illumination in the rendering phase.

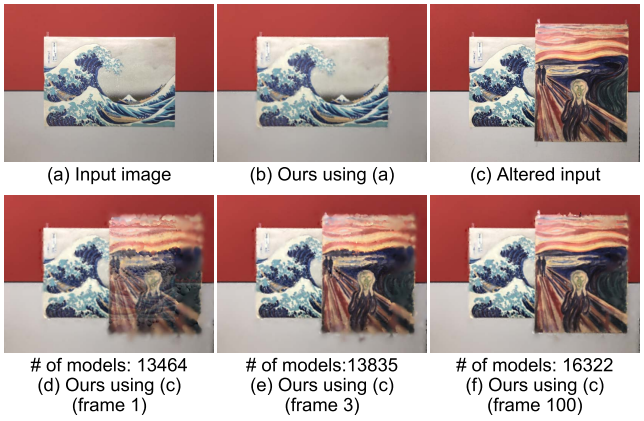


FIGURE 3. An example scenario for our online learning, where the input illumination (a) is partially changed to another (c). Our online learning approximates the input illumination closely and shows a nearly converged result (b). When a small part of the input changes (c), our method adapts the existing approximation for the changes (from (d) to (f)), without retraining models from scratch.

render virtual objects using the estimated radiance. Suppose we have n input samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where y_j is the j-th observed radiance (i.e., a pixel color) at $\mathbf{x}_j \in \Omega^R$ ($j \in \{1, \ldots, n\}$). We shall treat y_j as a one-dimensional value since we process each color channel independently and assume an additive noise model:

$$y_j = L_o^R(\mathbf{x}_j) + \epsilon_j, \tag{1}$$

where ϵ_j is a bounded error term whose expectation is zero, i.e., $E[\epsilon_j] = 0$. Intuitively, our learning problem is to construct an approximate function $\hat{L}_o^R(\mathbf{x})$, which is an estimate of the unknown $L_o^R(\mathbf{x})$, from the observed input samples, so that outgoing radiance at a query point $\mathbf{x}_q$ can be utilized in a rendering phase. To this end, we employ the *local regression* [6], [31] that locally approximates our target function, outgoing radiance, with multiple linear models. Let us denote the coefficients of the linear model at position $\mathbf{x}_c$ as $\boldsymbol{\beta}_c$ that includes the intercept and slopes (i.e., gradients) of the model. The linear model centered at $\mathbf{x}_c$ approximates the radiance in a neighbor position $\mathbf{x}_q$ near the $\mathbf{x}_c$, and an estimated radiance $\hat{y}_c(\mathbf{x}_q)$ using the c-th linear model is computed as

$$\hat{y}_c(\mathbf{x}_q) = \boldsymbol{\beta}_c^T \begin{bmatrix} 1 \\ \mathbf{x}_q \end{bmatrix}. \tag{2}$$

The coefficients of the model in the equation above are estimated by minimizing a weighted sum of squares $\sum_{j=1}^{n} w_{c,j} \left(y_j - \hat{y}_c(\mathbf{x}_j)\right)^2$. We define the weight $w_{c,j}$ allocated to the j-th sample as:

$$w_{c,j} = \exp\left(-\frac{||\mathbf{x}_j - \mathbf{x}_c||^2}{2b_c^2}\right), \tag{3}$$

where b_c is a bandwidth parameter that controls the size of the area covered by the model. We will present our optimization that adjusts the parameter per model in Sec. IV-B.

A query point $\mathbf{x}_q$, where we will infer an estimated radiance in AR rendering, can be covered by multiple linear models. As a result, we linearly interpolate the radiance values estimated from the neighboring models $\mathcal{N}(\mathbf{x}_q)$ of $\mathbf{x}_q$, to determine the final estimate:

$$\hat{L}_o^R(\mathbf{x}_q) = \frac{1}{W_c} \sum_{c \in \mathcal{N}(\mathbf{x}_q)} w_{c,q} \hat{y}_c(\mathbf{x}_q), \tag{4}$$

where W_c is the normalization term, $W_c = \sum_{c \in \mathcal{N}(\mathbf{x}_q)} w_{c,q}$. The estimated radiance $\hat{L}_o^R(\mathbf{x}_q)$ on a surface point $\mathbf{x}_q$ will be exploited in our AR rendering phase (Sec. V). Also, during our learning phase, we perform the interpolation (Eq. 4) at each pixel so that our intermediate results can be visualized to a user interactively.

The presented regression technique can be considered conceptually simple, but two major challenges exist for our problem. First, the model parameters $\boldsymbol{\beta}_c$ should be updated online since we aim to process streaming input from a hand-held device, and thus the number of samples n is a variable (not a fixed value). Second, the bandwidth parameter b_c should be adapted in a data-driven way so that its approximation error, e.g., a difference between $\hat{L}_o^R(\mathbf{x})$ and unknown $L_o^R(\mathbf{x})$, can be minimized. In the subsequent sections, we propose an online learning technique that exploits recursive least squares (RLS) [24] and online parameter learning [32] while tackling the technical challenges.

A. ONLINE CREATION AND UPDATE OF MODELS

This section explains how we create and update linear models in our online scenario, where a newly observed sample $(\mathbf{x}_{n+1}, y_{n+1})$ is given progressively, as illustrated in Fig. 4.

1) ONLINE CONSTRUCTION OF MODELS

As a model can only cover a local area near the model, we should construct multiple models so that the sampling domain Ω^R, where $L_o^R(\mathbf{x})$ at $\mathbf{x} \in \Omega^R$ is defined, can be fully covered. We create a new model when the existing

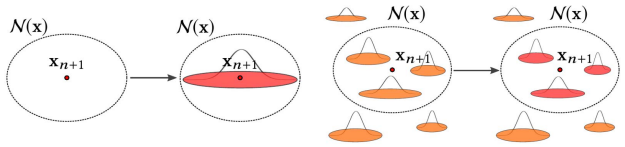


FIGURE 4. An illustrative figure for our model creation and update. Given a new sample at $\mathbf{x}_{n+1}$, we search neighbor models $\mathcal{N}(\mathbf{x}_{n+1})$ within a search radius (denoted as dashed circles). If there are no such models, we create a new model at $\mathbf{x}_{n+1}$ (a). Otherwise, we update the parameters of the neighboring models (b).

models cannot cover a new sample (i.e., $(n+1)$-th sample), analogously in [32]. Specifically, given a new sample $(\mathbf{x}_{n+1}, y_{n+1})$, we search models nearby $\mathbf{x}_{n+1}$ within a search radius (set to 0.2 meters) and check if their weights $w_{c,n+1}$ to the sample are larger than a predefined threshold $\epsilon = 0.1$. When we find the models that satisfy the condition, the respective models are updated using the new sample. We refer to the models as neighboring models $\mathcal{N}(\mathbf{x}_{n+1})$ of the new sample $\mathbf{x}_{n+1}$. If we fail to find a neighbor (i.e., $\mathcal{N}(\mathbf{x}_{n+1}) = \emptyset$), we create a new model at the point $\mathbf{x}_{n+1}$, as it indicates that the sample comes from an unobserved area.

2) LOCAL TRANSFORMATION

Before updating the neighboring models of a new sample, we apply a local transformation into the $\mathbf{x}_{n+1}$. Learning the model coefficients on a global space (i.e., a world coordinate) can be a straightforward option, but this approach can become inefficient for our problem since we can observe the data $\mathbf{x}_{n+1}$ on real surfaces (not in free space). Fig. 5 illustrates our local transformation that consists of translation, rotation, and projection. The local transformation varies per model, and thus we perform the transformation for all neighboring models $\mathcal{N}(\mathbf{x}_{n+1})$ and update the neighboring models independently with the transformed sample. Let us denote the transformed point by $\bar{\mathbf{x}}_{n+1} \in \mathbb{R}^2$. The weight function without the transformation (Eq. 3) is converted into a compact form:

$$w_{c,j} \equiv \exp\left(-\frac{||\bar{\mathbf{x}}_j||^2}{2b_c^2}\right). \tag{5}$$

3) ONLINE UPDATE OF MODEL COEFFICIENTS

Once we transform the new sample, we update the coefficients $\boldsymbol{\beta}$ of neighboring models $\mathcal{N}(\mathbf{x}_{n+1})$ using *recursive least squares* (RLS) [24]. We will drop the subscript c in the model coefficients (and other variables) for brevity since our update is performed for each model independently. Also, we add superscript $n+1$ into the coefficients (e.g., $\boldsymbol{\beta}^{n+1}$) to explicitly denote the updated values given the $(n+1)$-th sample. We adopt the following RLS update rules:

$$\mathbf{P}^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}^n - \frac{\mathbf{P}^n\tilde{\mathbf{x}}_{n+1}\tilde{\mathbf{x}}_{n+1}^T\mathbf{P}^n}{\lambda w_{n+1}^{-1} + \tilde{\mathbf{x}}_{n+1}^T\mathbf{P}^n\tilde{\mathbf{x}}_{n+1}}\right), \tag{6}$$

$$\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^n + w_{n+1}\mathbf{P}^{n+1}\tilde{\mathbf{x}}_{n+1}\left(y_{n+1} - (\boldsymbol{\beta}^n)^T\tilde{\mathbf{x}}_{n+1}\right). \tag{7}$$

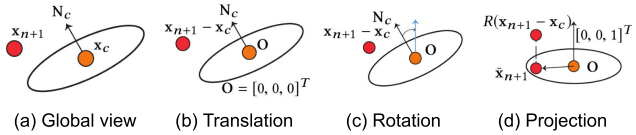


FIGURE 5. Illustration of a local transformation. We translate $\mathbf{x}_{n+1}$ to the local space where $\mathbf{x}_c$ is the origin (b), then the translated point is rotated with a rotation operator $R(\cdot)$ that maps a surface normal N_c at the point $\mathbf{x}_c$ into $[0, 0, 1]^T$ (c). Lastly, the transformed vector is projected onto a *xy* plane (d).

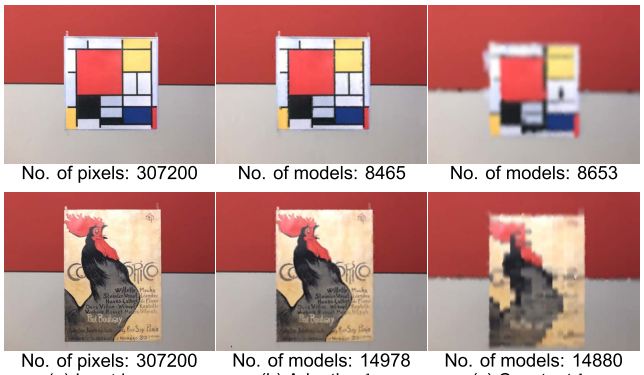


FIGURE 6. Visualization of the estimated illumination results using our adaptive bandwidth (b) and a constant bandwidth (c), given example input images (a). We adjust the constant bandwidths (i.e., globally fixed numbers) so that the numbers of its models become similar to those of our adaptive bandwidth selection. Our adaptive selection approximates high-frequency details in the input illumination (i.e., RGB images) much more accurately than the alternative.

where $\tilde{\mathbf{x}}_{n+1}^T = [1, \bar{\mathbf{x}}_{n+1}^T]$ and λ is a forgetting factor that is typically set to a number close to one. For example, we set λ to 0.97 initially and increase the value up to 0.9999 over time using an interpolation. $\mathbf{P}^{n+1}$ is an inverse covariance matrix of size 3×3.

For the recursion rule above, we set $\boldsymbol{\beta}^0$ and $\mathbf{P}^0$ to the zero vector and $10^5\mathcal{I}$ ($\mathcal{I}$ is the identity matrix) respectively when those are initialized, i.e., when a model is created. Note that this update is performed whenever we observe a sample, and thus this process is fully online.

B. BANDWIDTH OPTIMIZATION

While our online update rules enable us to estimate the model coefficients progressively, it requires allocating a proper weight w_i to the i-th sample. We will omit a model index c from variables (e.g., w_i=$w_{c,i}$) for brevity, as the bandwidth optimization is conducted per model independently. We present a local optimization that adjusts the bandwidth parameter b used in the weight function (Eq. 5) so that local models can approximate the radiance $L_o^R(\mathbf{x})$ closely. To this end, we define an objective function for our optimization as:

$$J \equiv \underbrace{\sum_{i=1}^{n+1}\frac{w_i}{W}\left\|y_i - \hat{y}_i(\bar{\mathbf{x}}_i)\right\|_2^2}_{J_1} + \underbrace{\gamma_1(\boldsymbol{\beta}^{n+1})^T(\boldsymbol{\beta}^{n+1})}_{J_2} + \underbrace{\gamma_2 b^{-4}}_{J_3}, \tag{8}$$

(a) I^B (b) I^R (c) I^{R+V} (d) I^F

FIGURE 7. An example result of our differential rendering that adds the difference between two rendering results ((b) and (c)) without and with a virtual object, into an observed background (a) so that a final AR rendering output (d) can be produced.

where $W = \sum_{i=1}^{n+1} w_i$. We seek to estimate the optimal bandwidth that minimizes the cost function, and the optimal will be used for the RLS update rules (Eq. 6 and 7).

In Eq. 8, J_1 is a weighted sum of the residual errors and J_2 is a regularization term that prevents the model coefficients from being too large. Also, we include a penalty term J_3 that prevents the bandwidth from becoming a very small value (and thus results in *overfitting*). γ_1 and γ_2 are the parameters that control the relative importance of J_2 and J_3 in the cost function above. We set those values to one and $1E-10$, respectively. We solve the optimization using a stochastic gradient descent [32] (see the supplemental report for details).

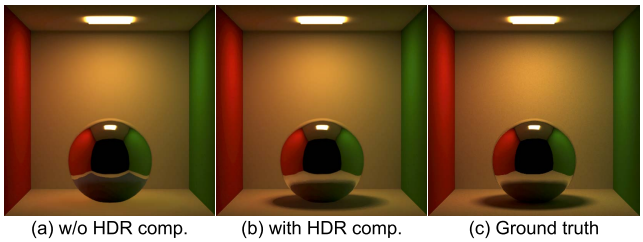
Fig. 6 shows an example result with and without our bandwidth optimization. As seen in the figure, the adaptive bandwidth selection allows our technique to preserve high-frequency details more accurately than a simple alternative that uses a constant bandwidth.

V. AR RENDERING USING ESTIMATED ILLUMINATION

Once our online learning generates a set of linear models (Sec. IV), we can estimate the radiance $\hat{L}_o^R(\mathbf{x}_q)$ at a query point $\mathbf{x}_q$. It allows us to exploit a global illumination technique that renders virtual objects using the estimated scene illumination. Our rendering is built upon the differential rendering framework [10], which combines a (vectorized) background image I^B (Fig. 7 (a)) (i.e., an image observed from a device) with a differential rendering result using virtual objects:

$$I^F = (\mathbf{1} - \mathbf{a}) \otimes I^B + \mathbf{a} \otimes \left(I^{R+V} - I^R \right), \tag{9}$$

where $\otimes$ is the element-wise multiplication and I^R (illustrated in Fig. 7) is a rendered image in a local region Ω^L around the space Ω^V where virtual objects are augmented. We separate the space Ω^R of a real scene into two disjoint spaces, local Ω^L and distant areas Ω^D, analogously in [10]. Specifically, we compute the bounding boxes of virtual models and enlarge the boxes slightly (e.g., 5 to 15 percent). Then, we define the points within the extended boxes as the local region Ω^L. $\mathbf{a}$ is a vector that contains alpha-masking values that determine blending factors between I^B and $(I^{R+V} - I^R)$. We generate primary rays from the camera per pixel and check whether the intersection points between the rays and the scene are within the local region Ω^L. Then, the alpha-masking value for a pixel is set to the fraction of how many rays intersect with the local area. For example, when the intersection points of all rays are within the region, the value is set to 1.



(a) w/o HDR comp. (b) with HDR comp. (c) Ground truth

FIGURE 8. Rendering results without and with our HDR compensation. We simulate our illumination learning and AR rendering from LDR samples (e.g., clamped colors) in a virtual space. The Cornell box is a simulated real scene, and the sphere is an augmented virtual object. Our compensation allows us to recover the illumination effect (the shadow below the object) lit by an HDR area light.

We assume that the radiance in the Ω^D is not affected by augmenting virtual objects. On the other hand, the radiance in the Ω^L should be amended since the virtual objects can change the illumination (e.g., shadows below the virtual object). Therefore we compute an additional rendering image I^{R+V} (Fig. 7 (c)) in the local area, including virtual objects. The rendering difference caused by augmenting virtual objects is added to the background image I^B to produce the final image I^F (Fig. 7 (d)).

A. HDR COMPENSATION AND REFLECTANCE ESTIMATION

Illuminating virtual objects through the differential rendering framework (Eq. 9) requires estimating the reflectance of surfaces in the local area Ω^L. We assume the reflectance is the same for all points in the Ω^L, similarly to [10], and denote it as ρ. Also, we should compensate for an energy loss in a saturated region of the observed image. Note that a hand-held device typically does not produce a high dynamic range (HDR) image, and thus the rendering results of virtual objects look darker than other objects in a real scene unless we compensate for the energy loss in saturated image regions (e.g., pixels on light sources). Specifically, we check all estimated models if a model is (potentially) saturated by testing whether its intercept term (i.e., the first element in the coefficients β_c) is higher than a threshold λ_{max} (e.g., 0.95) that is close to the maximum intensity (1.0 in our setting). For such models, we increase their intercept terms into $\lambda_{max} + \alpha_{loss}$.

As a result, we should estimate two global parameters, ρ and α_{loss}. A commonly adopted approach is to find the optimal parameters that minimize the difference between the observed intensities (e.g., I^B) and rendered results (e.g., I^R) while varying the parameters (e.g., [10], [37]). We also formulate this problem into a similar minimization problem:

$$\hat{\rho}, \hat{\alpha}_{loss} = \arg\min_{\rho, \alpha_{loss}} \sum_p \left| I_p^B - \frac{\pi\rho}{n_t} \sum_{j=1}^{n_t} \mathbb{F}\left(\hat{L}_i^R(\mathbf{x}_p, \omega_j) \right) (\omega_j \cdot N_p) \right|, \tag{10}$$

where the function $\mathbb{F}$ is defined as

$$\mathbb{F}(\cdot) = \begin{cases} \hat{L}_i^R(\mathbf{x}_p, \omega_j) & \text{if } \hat{L}_i^R(\mathbf{x}_p, \omega_j) < \lambda_{max} \\ \lambda_{max} + \alpha_{loss} & \text{otherwise.} \end{cases} \quad (11)$$

In Eq. 10, we compute the sum of the differences between observed intensities and rendering values by casting n_t primary rays. $\hat{L}_i^R(\mathbf{x}_p, \omega_j)$ is the estimated incident radiance at $\mathbf{x}_p$ with direction ω_j. Specifically, we compute the point $\mathbf{x}_p$ and normal N_p by casting a primary ray that passes the pixel position p from the camera origin. Also, we cast a ray from $\mathbf{x}_p$ with direction ω_j and find the intersection point $\mathbf{x}_q$ between the ray and scene. Then, we estimate the radiance using $\hat{L}_o^R(\mathbf{x}_q)$ at the query point $\mathbf{x}_q$ (Sec. IV).

We solve the minimization problem with constraints ($0 \leq \rho \leq 1$ and $\lambda_{max}+\alpha_{loss} \geq 1$) using a non-convex optimizer [1]. Note that we assume that a real scene is static, and thus we estimate the parameters only once when the positions of virtual objects are determined. In our experiments, the computational cost of estimating both parameters (ρ and α_{loss}) took approximately 0.2 secs when $n_t = 2048$. Fig. 8 shows a rendering result with and without our compensation for an energy loss. We produce the example figures using a virtual test since a ground truth image cannot be obtained in a real test. Specifically, we have employed a light transport algorithm, path tracing, in a physically-based rendering framework (PBRT [29]) for the test, which produces ground truth HDR radiance samples. We have clamped the HDR samples to generate LDR samples and fed the clamped samples to our method, and then generated local models using the LDR samples and computed their HDR compensation. As shown in the figure, the rendering result using our compensation is more accurate than the one without the compensation.

B. GLOBAL ILLUMINATION WITH ESTIMATED ILLUMINATION

Once the parameters ($\hat{\rho}$ and $\hat{\alpha}_{loss}$) are determined, our final task is to provide a realistic rendering result to a user at an interactive rate by adding the differential term ($I^{R+V} - I^R$) into the observed image I^B (see Eq. 9). To this end, we employ a general global illumination technique, path tracing [16]. To adopt the light transport algorithm, we should define the place where we can query for a radiance value $\hat{L}_o^R(\mathbf{x}_q)$, i.e., a weighted sum of linear models (Eq. 4). Note that we assume the illumination of a real scene does not change in the distance region Ω^D, and thus we perform the radiance query $\hat{L}_o^R(\mathbf{x}_q)$ only at $\mathbf{x}_q \in \Omega^D$.

Given the assumption, we generate multiple primary rays from the camera per pixel and trace those rays recursively until those rays intersect with the Ω^D. Note that we do not need to trace secondary rays when the intersection points of the rays are in the distance region Ω^D since the area's illumination is not affected by augmented virtual objects based on the assumption. We run this modified path tracing using our radiance estimation twice for generating I^R and I^{R+V}, which are rendering results without and with virtual objects. We then add the difference ($I^R - I^{R+V}$) into a background image I^B.

FIGURE 9. Interactive visualization results of our estimated illumination. Our method provides intermediate learning results to a user as live feedback during learning.

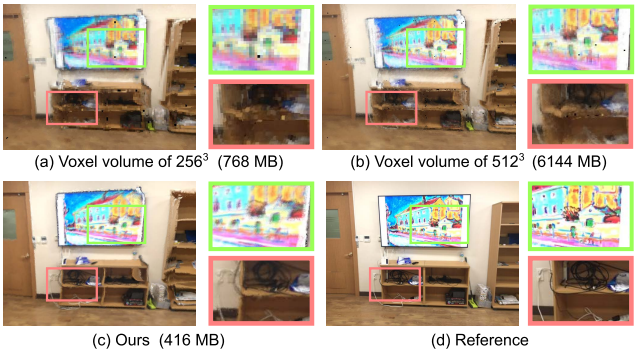


FIGURE 10. Visualization results of a voxel volume ((a) and (b)) and our estimated illumination (c). We train both methods using the same image sequence and show estimated illumination results at a specific frame. The voxel volume (a) shows discretization artifacts since it uniformly divides a given space. It is potentially possible to reduce the artifacts by increasing the granularity of volume (b), but it significantly increases its memory usage. On the other hand, our method (c) with a much smaller overhead approximates high-frequency details adequately.

As we aim to achieve an interactive rendering, we can use only small numbers of samples per pixel (e.g., two samples). As a result, our final image I^F can be corrupted by noise. To eliminate the noise, we apply a real-time denoiser [9] together with a temporal reprojection [2] into the noisy image I^F. We refer to a survey [39] that provides a comprehensive overview on denoising.

VI. IMPLEMENTATION DETAILS

We have implemented our online learning and rendering using CUDA and the Optix library [28].

A. HARDWARE CONFIGURATION

A server-client configuration, which communicates through a wireless network, has been used to implement our framework. We have used an iPad with a depth sensor (Structure Sensor MK[1]) to acquire RGBD images for the client. For the server where we perform our learning and rendering, we have used

[1] https://structure.io

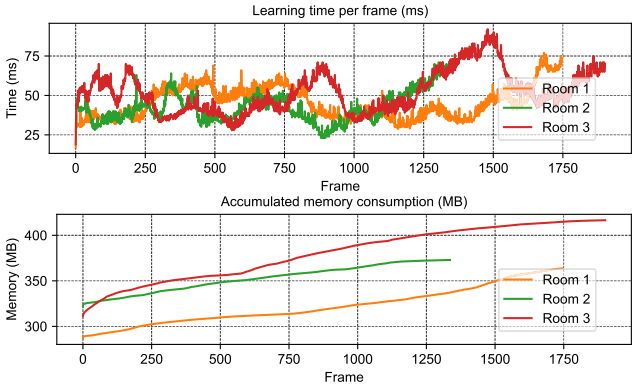


FIGURE 11. Computational and memory overheads for our online illumination learning for the scenes in Fig. 9.

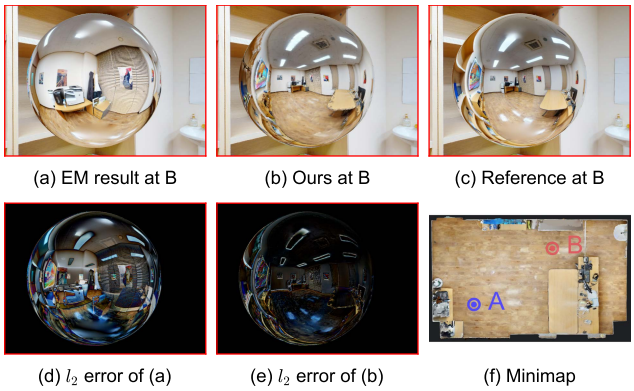


FIGURE 12. Comparisons with environment mapping (EM) that uses an environment map captured at a specific position (denoted by the blue circle *A* in (f)). EM can produce an accurate result when the virtual object is placed at the position where its map is captured, but it produces an erroneous result (a) when the object is augmented at a different position (the red circle *B* in (f)). On the other hand, our rendering output (b) is similar to the reference (c) that is rendered by EM, where we use the accurate map captured at the location of the virtual object. The visualized errors ((d) and (e)) are computed using the reference image (c).

a single desktop PC with an AMD threadripper and NVIDIA TITAN RTX.

B. GEOMETRIES OF A REAL SCENE

We have constructed geometries of a real scene before we learn the illumination of the scene, and the geometries are given to our learning and rendering as input. We have estimated the trajectory of the camera using the built-in SDK in the Structure Sensor and generated a triangular mesh using Open3D [38]. We observed that the reconstructed mesh could have erroneous holes caused by depth measurement failure on specular objects (e.g., glasses or mirrors). We manually filled the holes, analogously in [37]. As the geometries are prepared before running our algorithm, it is needed to align the camera pose in our illumination learning and rendering phase with the constructed mesh. To this end, we have built a SLAM map using the existing SLAM technique [20] and performed re-localization to identify the initial camera pose at the first frame. We then have used the Structure SDK to estimate a relative change of the camera pose for the other frames.

C. POINT CLOUD GENERATION IN RUNTIME

In our illumination learning phase, we take an RGBD input image where each pixel is converted into a color sample aligned with a world coordinate. It is straightforward to convert a depth value into the corresponding world coordinate, using the estimated camera pose from the Structure SDK. However, we observed that the computed world coordinates could be erroneous due to depth measurement errors. To mitigate this problem, we have instead generated a virtual ray from the estimated camera origin to each pixel center, then found an intersection point between the geometries and the ray, analogously in [37].

D. DATA STRUCTURE FOR THE NEAREST NEIGHBOR SEARCH

To find the nearest local models from a given point, we need an acceleration data structure that contains the models. We have used a uniform grid containing corresponding local models whose centers are within the grid.

VII. RESULTS

We have tested our online illumination learning for three indoor scenes (shown in Fig. 9) and generated AR rendering results by augmenting virtual objects with different materials into the scenes. We compare our illumination learning with a voxel-based representation that discretizes a given scene uniformly. Specifically, we have recorded all samples that our online method took as input and constructed the voxel volume where each voxel contains an averaged illumination value. Note that our online method does not need to store all samples. We store only local models and update the models progressively without keeping samples. We have given the same point cloud to the baseline for a fair comparison. We have also adjusted the volume resolution (i.e., the size of each regular volume) so that its memory footprint becomes higher than our method.

For rendering comparisons, we compare our AR rendering with the state-of-the-art [30], which uses an extended voxel cone tracing that additionally generates secondary cones to support glossy and specular materials. We have also used the clipmap texture [33] to represent its voxel volume efficiently. We also compare our method with a recent deep learning technique [22] using the public source code provided by the authors. We apply our post-processing (e.g., denoising) to their results for a fair comparison.

A. ONLINE LEARNING RESULTS WITH INTERACTIVE FEEDBACK

Fig. 9 shows both observed images and our estimated illumination for the three scenes. Note that our online learning allows visualizing illumination (i.e., our intermediate

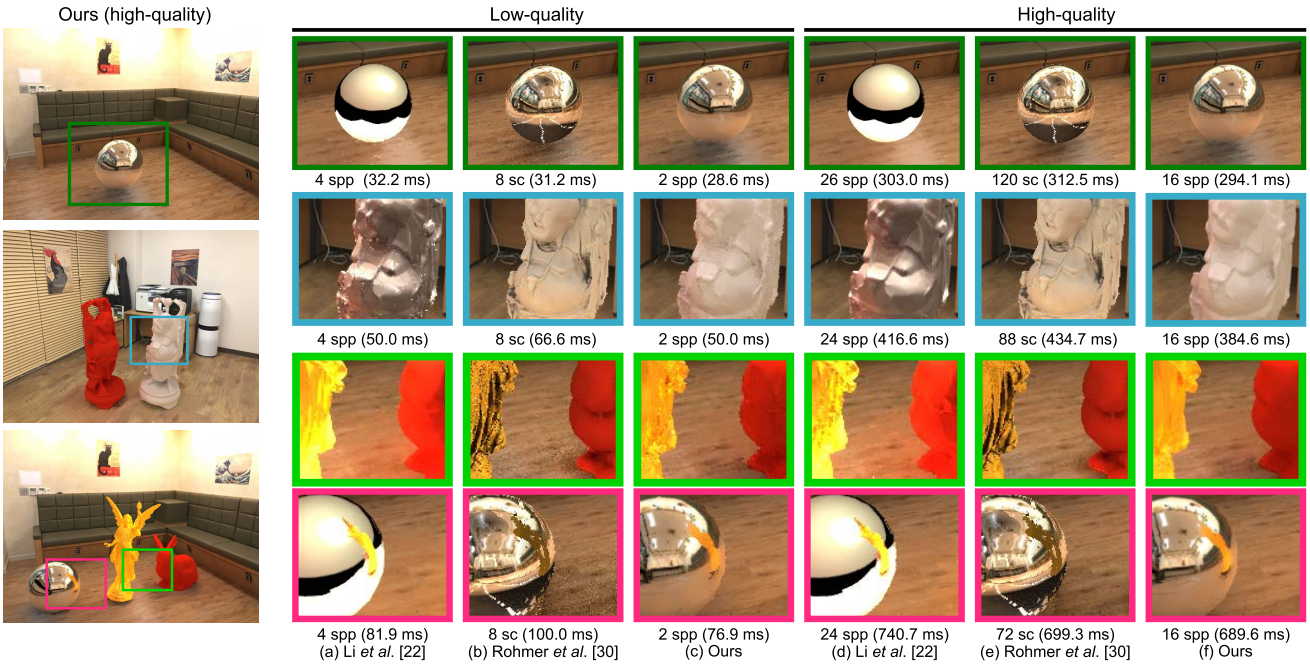


FIGURE 13. Equal-time comparisons with two recent methods, Li et al. [22] and Rohmer et al. [30] under low- and high-quality settings. We use diffuse materials for Buddha and Bunny models, a specular material (e.g., a mirror) for the Sphere, and a glossy material for the Lucy model. The learning-based method [22] does not capture reflected scene details on a mirrored sphere (in the top and bottom rows) and produces unnatural rendering results for the diffuse object (i.e.., the Buddha model). The existing SLAM-based method [30] preserves the reflected details but suffers from discretized artifacts caused by its voxel-based representation. Also, it does not handle the reflections on a glossy surface (i.e., the Lucy model) properly and produces noticeable artifacts. On the other hand, our method generates high-quality rendering results with much fewer visual artifacts, thanks to our high-quality representation of the scene illumination.

learning result at a frame) for a user during the learning process, unlike offline methods (e.g., [37]). The live feedback enables the user to identify problematic regions (blurry areas) to be refined. The accompanying video includes our interactive visualization results for the three scenes.

B. COMPARISONS OF ESTIMATED ILLUMINATION

In Fig. 10, we compare two estimated illumination results between our method and a voxel-based representation (i.e., a voxel volume). The figure shows that our method approximates high-frequency details adequately, even with lower memory usage, compared to the regular structure (voxel volume). Technically, our method allocates more models into high-frequency regions adaptively while maintaining a sparse number of models in low-frequency parts. The technical difference makes our method more memory-efficient and accurate than the well-known structure.

C. OVERHEAD OF OUR LEARNING

Fig. 11 shows our memory consumption and runtime overhead. Given the three test scenes, our learning times per frame are 45.0 ms, 41.8 ms, and 51.9 ms on average, respectively, and the visualization for the live feedback took approximately 5.0 ms. Additionally, the minimum and maximum learning times per frame are 17.0 ms and 77.4 ms for Room 1, 23.2 ms and 71.2 ms for Room 2, and 19.2 ms and 92.0 ms for Room 3, respectively. Our memory consumption increases over time as we dynamically create models, mainly when an input image contains an unobserved area. However, the increase rate in memory overhead becomes small as we go to the end of the capturing sequences. The final memory overheads at the end of the sequences are not significantly high (364 MB, 416 MB, and 373 MB for the scenes) if we consider the number of input samples (i.e., the total number of pixels from more than 1K input images). Our computational overheads for illumination learning can be higher than the voxel-based alternative [30] (e.g., 5 ms per frame given a 512^3 voxel resolution) since we dynamically create and update our local models, unlike the fixed approach. Nevertheless, our data-driven learning approximates the scene illumination more accurately, given a similar memory budget, as shown in Fig. 10.

D. ANALYSIS OF OUR SPATIALLY-VARYING ILLUMINATION

We analyze the spatially-varying nature of our learning process with the conventional environmental maps generated by measuring surrounding illumination at specific points using a physical device. We generated point clouds using a 3D scanner (Matterport Pro2 [5]). Note that we use a point cloud from a mobile device for all the other tests, and this test using a 3D scanner is used only for the analysis, which requires comparing our learned illumination and environmental maps at the same location. In Fig. 12, we render a virtual sphere using our estimated illumination and environmental mapping (EM).

(a) Voxel volume (b) Ours (c) Reference

FIGURE 14. Failure cases of the SLAM-based methods (voxel volume and ours). Both techniques show some misaligned artifacts in their estimated illumination due to erroneously estimated camera poses by a SLAM technique.

We capture environmental maps at predefined positions using the Matterport scanner to generate the EM results ((a) and (c) in Fig. 12). As shown in the figure, EM produces an erroneous result when a virtual object is augmented at a location different from where incident radiance is captured. On the other hand, our method approximates the scene illumination that varies spatially and produces a rendering output close to the reference.

E. EQUAL-TIME RENDERING COMPARISON

In Fig. 13, our method is compared to state-of-the-art techniques, Li et al. [22] and Rohmer et al. [30], given low- and high-quality settings. For the equal-time comparisons, we adjust the number of samples per pixel (spp) for both our method and Li et al. [22] and vary the number of secondary cones (sc) for Rohmer et al. [30].

The environmental mapping [22], which uses an estimated illumination by a neural network, can be a simple AR rendering option since it infers the map only from an observed single image. However, its rendering results are not accurate, especially when estimated scene illumination can be directly visible (e.g., reflected virtual objects on specular materials). Also, the method produces unnatural rendering results on a diffuse object (the Buddha model) while containing noticeable noise. Technically, it is challenging to correctly estimate the scene illumination since the technique infers an unobserved illumination from limited information (a single image).

The SLAM-based techniques (ours and Rohmer et al. [30]) produce more plausible results than the environmental mapping, as these methods exploit the observed (scanned) scene illumination in their illumination learning stages. Nonetheless, the previous technique shows discretization artifacts on the mirrored ball (in the top and bottom rows of the figure) caused by its voxel-based approximation for the scene illumination. Besides, it does not correctly handle the interreflections between virtual objects (e.g., see the visual artifacts on the Lucy model with a glossy material in the figure). On the other hand, our method captures the reflected high-frequency details (both real and virtual objects reflected on the specular object) appropriately without noticeable visual artifacts. Besides, our light representation allows us to use a full global illumination solution (i.e., path tracing) that accurately simulates the interreflections between virtual objects. It results in high-quality AR rendering when augmenting multiple virtual objects (e.g., the second and third scenes in the figure).

VIII. DISCUSSIONS AND FUTURE WORK

A. OUR ONLINE ILLUMINATION LEARNING

Our learning framework progressively approximates the illumination using a streaming RGBD sequence from a mobile device, allowing a user to refine the areas while showing intermediate learning results dynamically. The proposed online framework is able to produce a high-quality approximation of the scene illumination while maintaining interactive learning performance (less than 100 ms) per frame and moderate memory overheads (about 400 MB) for the tested indoor places.

Our online learning allows a user to easily update the pre-captured scene illumination since one can capture only the modified regions instead of conducting the whole learning process, as demonstrated in Fig. 3.

Our current framework, however, still requires capturing all necessary scene parts when the scene geometries are modified since we pre-compute the geometries of the scene (in Sec. VI). An ideal online framework would support a complete online learning process to perform the learning and AR rendering simultaneously. Still, a technically unresolved problem in our framework is handling dynamic geometries of the real scene. Extending our online learning and rendering system into a more comprehensive one, which supports complete dynamic scenarios, would be a challenging but important future research direction.

B. A TEXTURE-BASED ALTERNATIVE TO LOCAL MODELS

One may consider a texture-based approximation of the scene illumination, which can be an alternative to our local model-based approximation. For example, one can directly couple the reconstructed scene geometries with texture maps where a texel contains an illumination color at a specific position (e.g., a vertex of the scene mesh). While this approach can potentially approximate the illumination fruitfully when a sophisticated offline optimization is performed (e.g., [37]), it was unclear to perform such an approach for our online scenario where a streaming input is given.

Our framework exploits a uniform grid as a data structure for maintaining local models (in Sec. VI), but it requires a nontrivial time to search for nearest neighbor models from a query point. An interesting future work would be storing all local models into more efficient image-space data structures like texture maps once the learning process is complete since it can accelerate our AR rendering process.

C. LIMITATIONS AND FUTURE WORK

One technical limitation of our method, like other SLAM-based techniques, is that estimated illumination can be affected by errors in the camera pose predicted using a SLAM method, as shown in Fig. 14. The errors can be observed in final AR rendering results given a specific

scenario where reflected regions on virtual objects (e.g., a spherical mirror) correspond to the problematic region.

Our rendering framework supports an interactive AR experience but improving the rendering performance would be a fruitful research direction. For example, the estimated local models can be considered virtual point lights (VPLs) [17], and thus VPL-based acceleration schemes that handle many VPLs in a scalable manner (e.g., [23]) can be considered to improve the rendering performance further.

Another limitation of our method is that we assume the scene illumination is Lambertian. Thus it is not ideal for capturing radiances on shiny objects, which vary according to outgoing directions. Extending the local models into higher dimensional ones that fully approximate the directional changes of the scene radiances would also be future work.

IX. CONCLUSION

We have proposed a new illumination learning and rendering framework for a realistic AR. Our online learning approximates the illumination of an indoor scene with multiple local models, and the parameters (i.e., the coefficients and bandwidth parameter) of the models are updated progressively. The proposed optimization allows approximating the non-linear scene illumination closely even with low-order functions (i.e., linear models). Besides, our online learning allows users to check intermediate training results interactively while capturing the scene illumination. We have demonstrated that interactive rendering can be obtained using a differential rendering, which takes advantage of the estimated illumination using local models.

ACKNOWLEDGMENT

The authors thank the reviewers for the constructive comments and the Stanford Computer Graphics Laboratory for the virtual models (the David, Lucy, Bunny, and Buddha).

REFERENCES

[1] S. Agarwal, K. Mierle, and The Ceres Solver Team. (2022). *Ceres Solver*. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[2] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, andS. Hillaire, *Real-Time Rendering*, A. K. Peters, Ed., 4th ed. Boca Raton, FL, USA: CRC Press, 2018, ch. 12.2, pp. 522–523.

[3] A. Alhakamy and M. Tuceryan, "CubeMap360: Interactive global illumination for augmented reality in dynamic environment," in *Proc. SoutheastCon*, Apr. 2019, pp. 1–8.

[4] B. Burley and W. D. A. Studios, "Physically based shading at Disney," in *Proc. ACM SIGGRAPH Courses*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 1–7.

[5] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," in *Proc. Int. Conf. 3D Vis. (DV)*, Oct. 2017.

[6] S. W. Cleveland and C. Loader, "Smoothing by local regression: Principles and methods," in *Statistical Theory and Computational Aspects of Smoothing*, W. Härdle and G. M. Schimek, Eds. Berlin, Germany: Physica-Verlag, 1996, pp. 10–49.

[7] M. Corsini, M. Callieri, and P. Cignoni, "Stereo light probe," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 291–300, Apr. 2008.

[8] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive indirect illumination using voxel cone tracing," in *Proc. Symp. Interact. 3D Graph. Games (I D)*, Aug. 2011, p. 207.

[9] H. Dammertz, D. Sewtz, J. Hanika, and P. A. H. Lensch, "Edge-avoiding À-trous wavelet transform for fast global illumination filtering," in *Proc. Conf. High Perform. Graph. (HPG)*, 2010, pp. 67–75.

[10] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. New York, NY, USA: Association for Computing Machinery, May 2008, pp. 189–198.

[11] T. A. Franke, "Delta light propagation volumes for mixed reality," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2013, pp. 125–132.

[12] T. A. Franke, "Delta voxel cone tracing," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Sep. 2014, pp. 39–44.

[13] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagne, and J.-F. Lalonde, "Deep parametric indoor lighting estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2019, pp. 7174–7182.

[14] 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," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–14, Nov. 2017.

[15] M. Garon, K. Sunkavalli, S. Hadap, N. Carr, and J. Lalonde, "Fast spatially-varying indoor lighting estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019, pp. 6901–6910.

[16] J. T. Kajiya, "The rendering equation," *SIGGRAPH Comput. Graph.*, vol. 20. no. 4, pp. 143–150, Aug. 1986.

[17] A. Keller, "Instant radiosity," in *Proc. 24th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. Reading, MA, USA: ACM Press, 1997, pp. 49–56.

[18] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer, "Differential instant radiosity for mixed reality," in *Proc. 9th IEEE Int. Symp. Mixed Augmented Reality*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2010, pp. 99–107.

[19] J. Kronander, F. Banterle, A. Gardner, E. Miandji, and J. Unger, "Photorealistic rendering of mixed reality scenes," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 643–665, May 2015.

[20] M. Labbé and F. Michaud, "RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, 2019.

[21] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019, pp. 5911–5921.

[22] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and SVBRDF from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2475–2484.

[23] D. Lin and C. Yuksel, "Real-time stochastic lightcuts," *Proc. ACM Comput. Graph. Interact. Techn.*, vol. 3, no. 1, pp. 1–18, Apr. 2020.

[24] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA, USA: MIT Press, 1983.

[25] S. Ma, Q. Shen, Q. Hou, Z. Ren, and K. Zhou, "Neural compositing for real-time augmented reality rendering in low-frequency lighting environments," *Sci. China Inf. Sci.*, vol. 64, no. 2, pp. 1–15, Feb. 2021.

[26] M. Meilland, C. Barat, and A. Comport, "3D high dynamic range dense visual SLAM and its application to real-time object re-lighting," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2013, pp. 143–152.

[27] R. A. Newcombe, A. Fitzgibbon, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, and S. Hodges, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[28] G. S. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "OptiX: A general purpose ray tracing engine," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–13, Jul. 2010.

[29] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2016.

[30] K. Rohmer, J. Jendersie, and T. Grosch, "Natural environment illumination: Coherent interactive augmented reality for mobile and non-mobile devices," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 11, pp. 2474–2484, Nov. 2017.
[31] D. Ruppert and M. P. Wand, "Multivariate locally weighted least squares regression," *Ann. Statist.*, vol. 22, no. 3, pp. 1346–1370, Sep. 1994.
[32] S. Schaal and C. G. Atkeson, "Receptive field weighted regression," ATR Hum. Inf. Process. Laboratories, Kyoto, Japan, Tech. Rep. TR-H-209, 1997.
[33] C. C. Tanner, C. J. Migdal, and M. T. Jones, "The clipmap: A virtual mipmap," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, New York, NY, USA: Association for Computing Machinery, 1998, pp. 151–158.
[34] J. Unger, S. Gustavson, P. Larsson, and A. Ynnerman, "Free form incident light fields," in *Proc. 19th Eurographics Conf. Rendering (EGSR)*. Goslar, Germany: Eurographics Association, 2008, pp. 1293–1301.
[35] J. Unger, J. Kronander, P. Larsson, S. Gustavson, J. Löw, and A. Ynnerman, "Spatially varying image based lighting using HDR-video," *Comput. Graph.*, vol. 37, no. 7, pp. 923–934, Nov. 2013.
[36] J. Unger, A. Wenger, T. Hawkins, A. Gardner, and P. Debevec, "Capturing and rendering with incident light fields," in *Proc. 14th Eurographics Workshop Rendering (EGRW)*. Goslar, Germany: Eurographics Association, 2003, pp. 141–149.
[37] E. Zhang, M. F. Cohen, and B. Curless, "Emptying, refurnishing, and relighting indoor spaces," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–14, Nov. 2016.
[38] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*.
[39] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S.-E. Yoon, "Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 667–681, 2015.

WONJUN LEE received the B.S. degree in biomedical engineering from Yonsei University, in 2014, and the M.S. degree in electrical and computer engineering from Seoul National University, in 2018. He is currently pursuing the Ph.D. degree with the School of Integrated Technology, Gwangju Institute of Science and Technology. His research interests include rendering and augmented/virtual reality.

PILJOONG JEONG received the B.S. degree in computer engineering from Yonsei University, in 2017. He is currently pursuing the integrated M.S. and Ph.D. degrees with the School of Integrated Technology, Gwangju Institute of Science and Technology. His research interests include indoor scene understanding, SLAM and fusion-based geometry reconstruction, and differentiable rendering.

HAJIN CHOI received the B.S. degree from the School of Integrative Engineering, Chung-Ang University, in 2018. He is currently pursuing the integrated M.S. and Ph.D. degrees with the School of Integrated Technology, Gwangju Institute of Science and Technology. His research interests include rendering and augmented/virtual reality.

JINWOO KIM received the B.S. degree in electrical engineering and computer science from the Gwangju Institute of Science and Technology (GIST), in 2020. From 2019 to 2020, he worked as a Research Intern with the Computer Graphics Laboratory, GIST. He is currently working as an Assistant Manager with the Baggage Handling System Operations Team, Incheon International Airport Corporation. His research interests include deep learning and optimization on computer graphics and Monte Carlo simulation.

BOCHANG MOON (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the KAIST, in 2010 and 2014, respectively. He is currently an Associate Professor at the Gwangju Institute of Science and Technology (GIST). Before joining GIST, he was a Postdoctoral Researcher at Disney Research. His research interests include rendering, denoising, and augmented and virtual reality. He served as a PC Member for international conferences, including EGSR, I3D, PG, and CGI.