# Adaptive Irradiance Sampling for Many-Light Rendering of Subsurface Scattering

Kosuke Nabata, Kei Iwasaki

**Abstract**—Rendering a translucent material involves integrating the product of the transmittance-weighted irradiance and the BSSRDF over the surface of it. In previous methods, this spatial integral was computed by creating a dense distribution of discrete points over the surface or by importance-sampling based on the BSSRDF. Both of these approaches necessitate specifying the number of samples, which affects both the quality and the computation time for rendering. An insufficient number of samples leads to noise and artifacts in the rendered image and an excessive number results in a prohibitively long rendering time. In this paper, we propose an error estimation method for translucent materials in a many-light rendering framework. Our adaptive sampling can automatically determine the number of samples so that the estimated relative error of each pixel intensity is less than a user-specified threshold. We also propose an efficient method to generate the sampling points that make large contributions to the pixel intensity taking into account the BSSRDF. This enables us to use a simple uniform sampling, instead of costly importance sampling based on the BSSRDF. The experimental results show that our method can accurately estimate the error. In addition, in comparison with the previous methods, our sampling method achieves better estimation accuracy in equal-time.

**Index Terms**—subsurface scattering, BSSRDF, adaptive sampling, many-lights.

## 1 Introduction

Translucent materials such as marble, skin, wax, and so forth, exhibit characteristic soft appearance. This characteristic appearance comes from subsurface scattering effects that the incident light enters the surface of the translucent material, scatters multiple times inside it, and exits from different locations on the surface. The bidirectional scattering surface reflectance distribution function (BSSRDF) models the subsurface light transport between the point where the light enters and that where the light exits. Rendering translucent materials using BSSRDF involves the costly double integral, namely, the integral of the incident radiance over the hemisphere to compute the (transmittance-weighted) irradiance, and the spatial integral over the surface of the translucent material.

Scalable subsurface rendering methods [1], [2], [3] discretize the incident lighting into a huge number of virtual point lights (VPLs), and discretize the surface of the translucent material into sample points (we refer to these points as *irradiance points*). Then VPLs and irradiance points are clustered by constructing hierarchical structures of VPLs and irradiance points, to compute the double integral efficiently. Though efficient and practical, previous scalable rendering methods can suffer from errors due to clustering. Although these methods cluster VPLs and irradiance points so that the error of *each cluster* is bounded by a user-specified threshold, this does not guarantee that the total error from all the clusters is bounded by the threshold. As such, careful parameter tuning and re-renderings are imposed on the user to obtain rendering results with reliable accuracy.

One approach can control the total error from all the clusters for scalable subsurface rendering [4]. This method, however, shares the same problem of previous scalable approaches [1], [2], [3]. That is, the number of irradiance points is specified by the user before the rendering, though finding an appropriate number of irradiance points is a difficult task. An inappropriate number of irradiance points leads to noise and artifacts in the rendered images (Fig. 1(a)) despite the error estimation, since the true value, which is estimated using all VPLs and irradiance points, would be inaccurate. Using a large number of irradiance points would reduce the noise and artifacts, but it also increases the computation time (Fig. 1(b)). Importance-sampling the irradiance points based on the BSSRDF compounds this problem, since it involves costly ray-triangle intersections to generate the irradiance points, which accounts for most of the computation time.

To address these problems, we propose an adaptive sampling method of the irradiance points for the spatial integration of subsurface scattering. Unlike previous scalable approaches [1], [2], [3], [4] that prepare the irradiance points *prior to* the spatial integration computation, our method samples points *on-the-fly* so that the relative error of the estimated pixel intensity is smaller than a threshold based on a confidence interval [5]. To do so, our method constructs the hierarchical structures for the triangular meshes of translucent materials, instead of pre-sampled irradiance points. Then irradiance points to compute the spatial integral are sampled uniformly from the triangles. This enables us to increase the number of irradiance points by subdividing the triangles, which does not enforce on rebuilding the entire hierarchical structure. Moreover, we develop an efficient method for partitioning the triangular meshes taking into account the BSSRDF. This makes it possible to generate irradiance points that make large contributions to the pixel intensity, without requiring costly importance sampling. Thus our method achieves a significant speedup (up to

• *K. Nabata and K. Iwasaki are with the Department of Systems Engineering, Wakayama University, Wakayama, Japan.*
*E-mail: iwasaki@wakayama-u.ac.jp*


Fig. 1. (a) Using an inappropriate number of irradiance points leads to noise as shown in the insets in the red frames. To reduce noise, tweaking the number of irradiance points and re-renderings is often inevitable. Therefore, the total computation time needed to obtain the rendered results without noticeable noise might become large at all. (b) Using a large number of irradiance points wastes computation time in generating the irradiance points. (c) Our method automatically determines the number of irradiance points. Our method can yield rendering results without trial-and-error processes and re-renderings.

12x) compared to the previous methods [1], [4] in equal-quality rendering. While we introduce our sampling method in a many-light rendering framework, we also discuss the possibility of adapting our sampling method to path tracing in Sec. 5.4.

## 2 Related Work

In this section, we mainly focus on reviewing the previous work relevant to ours: many-light rendering and a hierarchical approach for the dipole model [6]. Although our method would work for other BSSRDF models if the upper bound of the BSSRDF between the points where the incident light enters and those where the scattered light exits can be easily obtained, incorporating sophisticated BSSRDF models (e.g., [7], [8], [9], [10], [11]) into our method is something planned for future work.

### 2.1 Many-light rendering

Many-light rendering methods approximate the computation of complex illumination by the computation of direct illumination from a set of virtual point lights (VPLs) [12], [13], [14]. Since many-light rendering often uses a huge number of VPLs to render realistic images, scalable approaches that cluster similar VPLs are widely used. Lightcuts [15] and its variants [1], [16], [17] are a scalable solution to many-light methods using a hierarchical representation of VPLs. These methods cluster VPLs so that the error in *each* cluster is bounded by a user-specified error threshold. Another scalable approach is formulating many-light rendering as a matrix sampling problem [18], [19], [20], [21]. Although these methods can significantly accelerate many-light rendering, they do not estimate *the total error from all the clusters*, which can emerge as noise and artifacts in the rendered images. As such, to obtain rendered images with reliable accuracy, these methods impose parameter tunings and re-renderings on the user. To address this problem, Nabata et al. proposed an error estimation framework for many-light rendering [22]. Although this method can be used to control the total error, it does not handle subsurface scattering of light.

Instead of clustering, a vast set of sensor/light pairs has been stochastically evaluated in the context of bidirectional path tracing [23], [24]. These methods, however, do not deal with subsurface rendering.

### 2.2 Spatial integration for subsurface rendeirng

Previous methods for computing the spatial integral can be categorized into two groups: preparing a dense distribution of irradiance points over the entire surface of a translucent material before rendering (referred to as *point cloud methods*), and sampling the irradiance points based on the BSSRDF (referred to as *importance sampling methods*).

#### 2.2.1 Point cloud methods

Point cloud methods densely distribute the irradiance points across the surface of a translucent material before rendering, and reuse the irradiance for all the pixels. Jensen and Buhler proposed a hierarchical two-pass approach to render translucent materials [25]. Arbree et al. proposed a single-pass approach using VPLs [2]. This method clusters the triples of VPLs, points to be shaded, and irradiance points. Wu et al. formulated calculation of the radiance for subsurface scattering as a matrix sampling problem [3]. Recently, Milaenen et al. proposed a dual hierarchy method combined with frequency analysis of the BSSRDF for efficient computation of the integral [26]. Although the point cloud methods can amortize the cost to compute the irradiance by reusing it, these methods require additional memory to store the irradiance points, which can be prohibitive especially for optically dense translucent materials (i.e., large extinction coefficient), since the point cloud methods often use the mean free-path (i.e., the reciprocal of the extinction coefficient) of the translucent material as the maximum distance between the irradiance points. In addition, when only a small fraction of the surface of a translucent material is rendered, densely distributing the irradiance points across the entire surface can be inefficient.

#### 2.2.2 Importance sampling methods

Importance-sampling of the irradiance points based on the BSSRDF addresses the problems of the point cloud methods. By exploiting the characteristics of the BSSRDF that decays exponentially, importance sampling methods [4], [16] generate the irradiance points in the proximity of a point to be shaded. Walter et al. incorporate the importance sampling of the BSSRDF into the VPL-based methods (i.e., Lightcuts) [16]. Although this method can render the translucent materials efficiently by retaining the merits of Lightcuts, this method also inherits the limitation of Lightcuts (i.e., the total error from all the clusters is not bounded). Sakai et al. extended the error estimation method [22] to render translucent materials [4]. Although the total error from all the clusters can be controlled using this method, it inherits the problems of importance sampling methods. That is, an inappropriate number of importance-sampled points leads to noise and artifacts in the rendered images despite estimating the errors, and the computational cost for importance-sampling of the irradiance points is high since it involves the costly ray-triangle intersections.

Our sampling method retains the merits of the importance sampling methods over the point cloud methods (i.e., avoiding wasteful sampling and the additional memory), but bypasses the costly ray-traingle intersections to sample the irradiance points. More importantly, both point cloud methods and importance sampling methods lack the ability to refine the irradiance points so as to improve the image quality. Since both methods represent the irradiance points with a hierarchical structure, refining the irradiance points involves rebuilding the hierarchical structure. Our method adaptively and automatically samples the irradiance points so that the estimated relative error is less than the user-specified threshold.

## 3 Background

### 3.1 Subsurface Rendering with a Diffuse BSSRDF

The BSSRDF is used to model subsurface scattering of light in rendering translucent materials. The outgoing radiance $L$ at point $\mathbf{x}_o$ in direction $\boldsymbol{\omega}_o$ is calculated by integrating the product of the BSSRDF, $S$, and the incident radiance, $L$, at point $\mathbf{x}_i$ over the surface $A$ of the translucent material and the hemisphere $\Omega$ as:

$$L(\mathbf{x}_o, \boldsymbol{\omega}_o) = \int_A \int_\Omega L(\mathbf{x}_i, \boldsymbol{\omega}_i) S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o)(\mathbf{n}_i \cdot \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i dA(\mathbf{x}_i),$$

where $\mathbf{n}_i$ is the normal at point $\mathbf{x}_i$. The BSSRDF $S$ can be represented by the sum of two terms, a single scattering term $S^1$ and a multiple scattering term $S^d$. The multiple scattering term $S^d$ is efficiently calculated from the diffuse BSSRDF $R_d$ [6]. Our method mainly focuses on calculating the outgoing radiance $L^d$ due to the diffuse BSSRDF. $L^d$ is given by:

$$L^d(\mathbf{x}_o, \boldsymbol{\omega}_o) = \frac{F_t(\boldsymbol{\omega}_o)}{\pi} \int_A R_d(r_i) \left( \int_\Omega L(\mathbf{x}_i, \boldsymbol{\omega}_i) F_t(\boldsymbol{\omega}_i)(\mathbf{n}_i \cdot \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i \right) dA(\mathbf{x}_i),$$

where $F_t$ is the Fresnel transmittance and $r_i = \|\mathbf{x}_i - \mathbf{x}_o\|$. Hereinafter, the point $\mathbf{x}_o$ used in computing the outgoing radiance $L^d$ is referred to as the *shading point*. Since the outgoing direction $\boldsymbol{\omega}_o$ is uniquely determined by the shading point $\mathbf{x}_o$ and the camera position, we drop $\boldsymbol{\omega}_o$ from $L^d(\mathbf{x}_o, \boldsymbol{\omega}_o)$ and the Fresnel transmittance $F_t(\boldsymbol{\omega}_o)$ is replaced with $F_t(\mathbf{x}_o)$ for the notation clarity.

### 3.2 Many-light Methods for Subsurface Rendering

Many-light methods represent the incident radiance $L(\mathbf{x}_i, \boldsymbol{\omega}_i)$ at the irradiance point $\mathbf{x}_i$ using virtual point lights (VPLs). By using a set of VPLs $\mathbb{L}$, the outgoing radiance $L^d$ is calculated by means of:

$$L^d(\mathbf{x}_o) \approx \frac{F_t(\mathbf{x}_o)}{\pi} \int_A R_d(r_i) \left( \sum_{\mathbf{y} \in \mathbb{L}} I(\mathbf{y}) H(\mathbf{y}, \mathbf{x}_i) \right) dA(\mathbf{x}_i),$$

where $I(\mathbf{y})$ is the intensity of the VPL $\mathbf{y}$. $H(\mathbf{y}, \mathbf{x}_i)$, which encompasses the geometry term $G$, the visibility function $V$, and the Fresnel transmittance $F_t$ at the irradiance point $\mathbf{x}_i$, is given by:

$$H(\mathbf{y}, \mathbf{x}_i) = G(\mathbf{y}, \mathbf{x}_i) V(\mathbf{y}, \mathbf{x}_i) F_t\left(\omega_{\mathbf{y}\mathbf{x}_i}\right),$$

where $\omega_{\mathbf{y}\mathbf{x}_i}$ is the unit direction from the irradiance point $\mathbf{x}_i$ to the VPL $\mathbf{y}$ as $\omega_{\mathbf{y}\mathbf{x}_i} = (\mathbf{y} - \mathbf{x}_i)/\|\mathbf{y} - \mathbf{x}_i\|$.

The pixel intensity for subsurface scattering of light is calculated by integrating the product of the importance function $W$ and $L^d$ over the pixel footprint $\mathcal{P}$.To calculate the integral over $\mathcal{P}$, the pixel footprint $\mathcal{P}$ is discretized into a set of points, $\mathbb{S}$, using supersampling. The pixel intensity $\mathcal{I}$ with supersampling is calculated as:

$$\mathcal{I} = \sum_{\mathbf{x}_o \in \mathbb{S}} \int_A w(\mathbf{x}_o) R_d(r_i) \left( \sum_{\mathbf{y} \in \mathbb{L}} I(\mathbf{y}) H(\mathbf{x}_i, \mathbf{y}) \right) dA(\mathbf{x}_i), \quad (1)$$

where $w(\mathbf{x}_o)$ is the weighting function that encompasses the importance function $W$ and the Fresnel transmittance $F_t$ and is given by:

$$w(\mathbf{x}_o) = \frac{W(\mathbf{x}_o) F_t(\mathbf{x}_o)}{\pi p(\mathbf{x}_o) |\mathbb{S}|}, \quad (2)$$

where $p$ is the sampling pdf of $\mathbf{x}_o$.

### 3.3 Error Estimation using a Confidence Interval

To make an estimate of some quantity (e.g., pixel intensity $\mathcal{I}$) using sampling, it is beneficial to assess the accuracy of the estimate, i.e., the error $|\mathcal{I} - \hat{\mathcal{I}}|$, where $\hat{\mathcal{I}}$ and $\mathcal{I}$ are the estimator and the true value, respectively. The confidence interval $\left[\hat{\mathcal{I}} - \Delta\mathcal{I}, \hat{\mathcal{I}} + \Delta\mathcal{I}\right]$ contains the true value $\mathcal{I}$ with a certain probability $\alpha$ as:

$$P_r(|\mathcal{I} - \hat{\mathcal{I}}| \leq \Delta\mathcal{I}) = \alpha, \quad (3)$$

where $\alpha$ is referred to as the *confidence level*, and $\Delta\mathcal{I}$ is calculated by:

$$\Delta\mathcal{I} = t_\alpha \sqrt{\frac{s^2}{n}},$$

where $t_\alpha$ is the $\alpha$ quantile of the Student $t$-distribution $t(x)$ that satisfies $\int_{-t_\alpha}^{t_\alpha} t(x)dx = \alpha$, $s^2$ is the sample variance of the estimator $\hat{\mathcal{I}}$, and $n$ is the number of samples. Note that $\Delta\mathcal{I}$ can be calculated by using only the sample variance and $t_\alpha$ *without* evaluating the true value $\mathcal{I}$. By specifying the confidence level $\alpha$ and the relative error threshold $\epsilon$, the error $|\mathcal{I} - \hat{\mathcal{I}}|$ is bounded by $\Delta\mathcal{I} < \epsilon\hat{\mathcal{I}}$ with probability $\alpha$.

In the context of scalable many-light rendering, the sampling domain is partitioned into subdomains (clusters). By partitioning the sampling domain into $J$ clusters, $\Delta\mathcal{I}$ is rewritten as [22]:

$$\Delta\mathcal{I} = t_\alpha \sqrt{\frac{N}{N - J} \sum_{j=1}^{J} \frac{s_j^2}{n_j}}, \quad (4)$$

where $s_j^2$ and $n_j$ are the sample variance and the number of samples of the $j$-th cluster. $N$ is the total number of samples $N = \sum_{i=1}^{J} n_j$, $n_j$ is allocated to $j$-th cluster based on Neymann allocation, which is the optimal method for allocating samples to each cluster with a fixed total number of samples $N$.

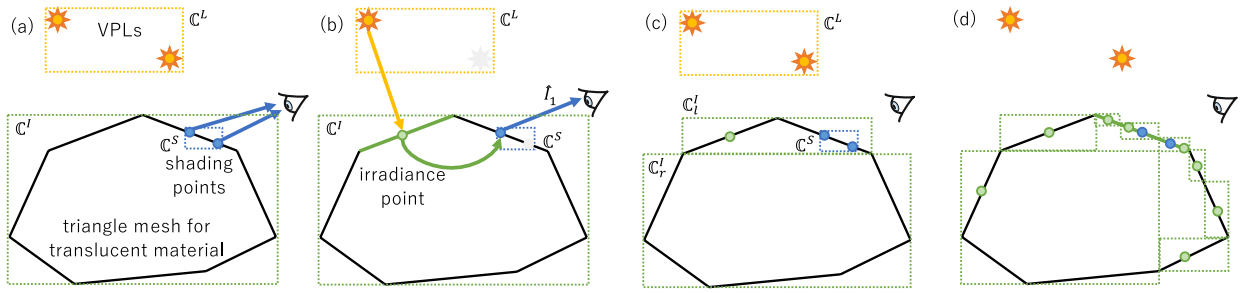


Fig. 2. Overview of our method. (a) shows the triplet $(\mathbb{C}^L, \mathbb{C}^I, \mathbb{C}^S)$. Unlike previous scalable approaches that cluster irradiance points, our method clusters the triangular mesh for a translucent material. (b) shows the estimate of the pixel intensity $\hat{\mathcal{I}}_1$ by sampling a VPL, irradiance point, and shading point from the triplet. To sample irradiance points from $\mathbb{C}^I$, we first sample a triangle and then an irradiance point is sampled from it. (c) shows the subdivision of the triplet $(\mathbb{C}^L, \mathbb{C}^I, \mathbb{C}^S)$ into two triplets $(\mathbb{C}^L, \mathbb{C}^I_l, \mathbb{C}^S)$ and $(\mathbb{C}^L, \mathbb{C}^I_r, \mathbb{C}^S)$ by subdividing IC $\mathbb{C}^I$ into $\mathbb{C}^I_l$ and $\mathbb{C}^I_r$. (d) when IC that contains one triangle is selected to be subdivided, our method subdivides the triangle into two triangles, and two ICs that contain each subdivided triangle are created. This enables us to generate irradiance points adaptively in the proximity of shading points, without the reconstruction of BVH (just replacing the leaf node with an inner node with two child nodes).

## 4 Our Approach

### 4.1 Algorithm Overview

Our goal is to estimate the intensity $\mathcal{I}$ of each pixel due to subsurface scattering of light by adaptively sampling the irradiance points so that the relative error of the estimate $\hat{\mathcal{I}}$ is less than a user specified threshold $\epsilon$. To this end, our method builds upon the previous clustering based approach proposed by Arbree et al. [2]. Our method estimates the pixel intensity in Eq. (1) by partitioning the integral domain $A$, the set of VPLs $\mathbb{L}$, and that of the shading points $\mathbb{S}$ into clusters. Each cluster is labeled either as an *IC (irradiance cluster)* $\mathbb{C}^I$, a *LC (light cluster)* $\mathbb{C}^L$, or a *SC (shading cluster)* $\mathbb{C}^S$, respectively. The combination of three clusters $(\mathbb{C}^L, \mathbb{C}^I, \mathbb{C}^S)$ is referred to as *triplet*. The pixel intensity $\mathcal{I}$ is estimated by sampling irradiance points, VPLs, and shading points from each cluster of the triplets.

The technical novelties of our approach are twofold. Firstly, our method adaptively subdivides the integral domain $A$ of the surface using triangular meshes instead of using pre-sampled points, as in the previous methods [2], [4]. This enables us to generate irradiance points on-the-fly and determine the number of the irradiance points automatically, freeing the user from specifying this. Secondly, we introduce an efficient method to partition the entire domain ($\mathbb{L} \times A \times \mathbb{S}$ in Eq. (1)) into the triplets taking into account the BSSRDF. In contrast to the previous methods [4], in which a cluster with the maximum extent is selected for subdivision, our method selects a cluster so as to reduce the variance of the estimate $\hat{\mathcal{I}}$. This enables us to sample irradiance points with large contributions without excessive subdivision of the surface or requiring costly importance sampling, resulting in a significant speedup.

Fig. 2 presents an overview of our method. ICs are represented by a bounding volume hierarchy (BVH) of triangles. The irradiance points are directly sampled from the triangles. The LCs and SCs are represented by binary trees similar to Multidimensional Lightcuts [1]. The binary tree for the LC and the BVHs for translucent materials are prepared and shared for all the pixels. For each pixel, we first generate shading points on the pixel footprint using supersampling and cluster shading points using a binary tree.

We start from the triplet comprising each root node (Fig. 2(a)). The estimate of the pixel intensity $\hat{\mathcal{I}}$ is calculated by sampling VPLs, irradiance points, and shading points from the triplet. The sample variance is also calculated to estimate the error $\Delta\hat{\mathcal{I}}$ (Sec. 4.2.2). Then the following processes are repeated:

1) Select the triplet $(\mathbb{C}^L_j, \mathbb{C}^I_j, \mathbb{C}^S_j)$ with the largest error bound $e_j$ (Sec. 4.2.3).
2) Select one cluster out of the three clusters $\mathbb{C}^L_j, \mathbb{C}^I_j, \mathbb{C}^S_j$ (Sec. 4.3) and subdivide it. Then the triplet is subdivided into two triplets (Fig. 2(c)).
3) For the subdivided two triplets, the estimates of the pixel intensities as well as the sample variance are calculated.
4) Update the estimated pixel intensity $\hat{\mathcal{I}}$ and the error $\Delta\hat{\mathcal{I}}$ using the sample variance.
5) Terminate if the estimated error $\Delta\hat{\mathcal{I}}$ is less than $\epsilon\hat{\mathcal{I}}$. Return to Step 1, otherwise.

### 4.2 Error estimation for the pixel intensity $\hat{\mathcal{I}}$

The pixel intensity $\mathcal{I}$ in Eq. (1) is calculated by partitioning the entire domain into each cluster. The pixel intensity $\mathcal{I}_j$ due to the $j$-th triplet $(\mathbb{C}^L_j, \mathbb{C}^I_j, \mathbb{C}^S_j)$ is calculated by simply replacing the entire domain in Eq. (1) with each cluster as:

$$\mathcal{I}_j = \sum_{\mathbf{x}_o \in \mathbb{C}^S_j} w(\mathbf{x}_o) \int_{A_j} R_d(r_i) \left( \sum_{\mathbf{y} \in \mathbb{C}^L_j} I(\mathbf{y}) H(\mathbf{y}, \mathbf{x}_i) \right) dA(\mathbf{x}_i),$$

where $A_j$ is the sum of the areas of the triangles in IC $\mathbb{C}^I_j$.

To estimate $\mathcal{I}_j$, our method samples irradiance points, VPLs, and shading points as:

$$\hat{\mathcal{I}}_j = \frac{1}{n_j} \sum_{k=1}^{n_j} \frac{w(\mathbf{x}_{o,k}) R_d(\|\mathbf{x}_{i,k} - \mathbf{x}_{o,k}\|) I(\mathbf{y}_k) H(\mathbf{y}_k, \mathbf{x}_{i,k})}{p_I(\mathbf{x}_{i,k}) p_S(\mathbf{x}_{o,k}) p_L(\mathbf{y}_k)}, \tag{5}$$

where $\mathbf{x}_{i,k}$, $\mathbf{x}_{o,k}$ and $\mathbf{y}_k$ are the $k$-th samples of irradiance points, shading points, and VPLs, respectively. VPLs and shading points are sampled proportional to their intensities

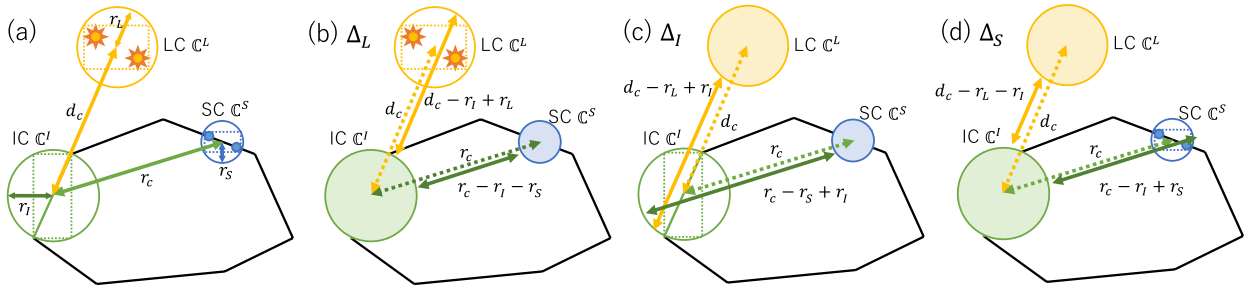


Fig. 3. Computation of $\Delta_L$, $\Delta_I$, and $\Delta_S$. (a) our method selects one cluster from three clusters $\mathbb{C}^L, \mathbb{C}^I$, and $\mathbb{C}^S$ so that the subdivision of selected cluster can reduce the variance of $R_d F$ most. To do so, our method uses the range of $R_d/d^2$ as the criterion (i.e., we consider the cluster with the largest range of $R_d/d^2$ as the largest variance). (b) computation of $\Delta_L$. The minimum/maximum distances between $\mathbb{C}^L$ and $\mathbb{C}^I$ (fixed) are $d_c - r_I - r_L$ and $d_c - r_I + r_L$. Since $\mathbb{C}^I$ and $\mathbb{C}^S$ are fixed in computing $\Delta_L$, the mimimum/maximum distances between $\mathbb{C}^I$ and $\mathbb{C}^S$ are fixed as $r_c - r_I - r_S$. $\Delta_I$ and $\Delta_S$ can be computed in the similar way as shown in (c) and (d).

or the weighting function values by using the following probability mass functions:

$$p_L(\mathbf{y}_k) = \frac{I(\mathbf{y}_k)}{\sum_{\mathbf{y}\in\mathbb{C}_j^L} I(\mathbf{y})} = \frac{I(\mathbf{y}_k)}{I_j},$$
$$p_S(\mathbf{x}_{o,k}) = \frac{w(\mathbf{x}_{o,k})}{\sum_{\mathbf{x}_o\in\mathbb{C}_j^S} w(\mathbf{x}_o)} = \frac{w(\mathbf{x}_{o,k})}{W_j},$$

where $W_j$ and $I_j$ are the sums of the weighting function values and the intensities over the clusters $\mathbb{C}_j^S$ and $\mathbb{C}_j^L$, respectively. Substituting these functions into Eq. (5) leads to the following equation:

$$\hat{\mathcal{I}}_j = \frac{W_j I_j}{n_j} \sum_{k=1}^{n_j} \frac{R_d(\|\mathbf{x}_{i,k} - \mathbf{x}_{o,k}\|) H(\mathbf{y}_k, \mathbf{x}_{i,k})}{p_I(\mathbf{x}_{i,k})}. \quad (6)$$

#### 4.2.1 Sampling irradiance points

Our method samples $n_j$ irradiance points uniformly from the cluster $\mathbb{C}_j^I$. If the cluster $\mathbb{C}_j^I$ contains $T_j$ triangles $\{t_1, \ldots, t_{T_j}\}$ with areas $\{a_1, \ldots, a_{T_j}\}$, we first sample a triangle in proportion to its area, then an irradiance point is uniformly sampled from the triangle. That is, the probability density function (pdf) of the sample irradiance points is:

$$p_I(\mathbf{x}_{i,k}) = \frac{a}{\sum_{l=1}^{T_j} a_l} \cdot \frac{1}{a} = \frac{1}{\sum_{l=1}^{T_j} a_l} = \frac{1}{A_j}, \quad (7)$$

where $A_j$ is the sum of the areas of the triangles in $\mathbb{C}_j^I$.

#### 4.2.2 Estimate of $\hat{\mathcal{I}}$ and $\Delta\hat{\mathcal{I}}$

By substituting the pdf $p_I(\mathbf{x}_{i,k})$ into Eq. (6), the pixel intensity $\hat{\mathcal{I}}_j$ due to the $j$-th triplet is calculated using:

$$\hat{\mathcal{I}}_j = \frac{A_j W_j I_j}{n_j} \sum_{k=1}^{n_j} R_d(\|\mathbf{x}_{i,k} - \mathbf{x}_{o,k}\|) H(\mathbf{y}_k, \mathbf{x}_{i,k}). \quad (8)$$

The estimate of the pixel intensity $\hat{\mathcal{I}}$ is calculated by summing $\hat{\mathcal{I}}_j$ over all the triplets as $\hat{\mathcal{I}} = \sum_{j=1}^{J} \hat{\mathcal{I}}_j$ where $J$ is the number of all triplets. Following a previous method for estimating errors [22], our method uses $n_j = 2$ samples (thus the total number of samples $N = \sum_{j=1}^{J} n_j = 2J$). The estimate of the error $\Delta\hat{\mathcal{I}} = |\hat{\mathcal{I}} - \mathcal{I}|$ in Eq. (4) is simplified to:

$$\Delta\hat{\mathcal{I}} = t_\alpha \sqrt{\frac{2J}{2J - J} \sum_{j=1}^{J} \frac{s_j^2}{2}} = t_\alpha \sqrt{\sum_{j=1}^{J} s_j^2}, \quad (9)$$

where $t_\alpha$ is the $\alpha$ quantile of the $t$-distribution and $s_j^2$ is the sample variance of $\hat{\mathcal{I}}_j$ for the $j$-th triplet as:

$$s_j^2 = \frac{A_j^2 W_j^2 I_j^2}{2} (R_d(r_1) H(\mathbf{y}_1, \mathbf{x}_{i,1}) - R_d(r_2) H(\mathbf{y}_2, \mathbf{x}_{i,2}))^2,$$

where $r_1 = \|\mathbf{x}_{i,1} - \mathbf{x}_{o,1}\|$ and $r_2 = \|\mathbf{x}_{i,2} - \mathbf{x}_{o,2}\|$, respectively.

#### 4.2.3 Triplet selection using the error bound

To reduce the error $\Delta\hat{\mathcal{I}}$, our method selects the triplet with the maximum error bound and subdivides it. Although the standard deviation of each triplet can be used as the error bound, the computational cost of the standard deviation is high since it involves costly evaluations of $R_d H$ in Eq. (8) over all VPLs, irradiance points, and shading points in the triplet. Instead, our method calculates the error bound $e_j$ for the $j$-th triplet as follows (the derivation is shown in Appendix A.):

$$e_j = \frac{A_j W_j I_j}{2\sqrt{n_j}} R_d^u G^u F_t^u,$$

where $R_d^u$, $G^u$, and $F_t^u$ are the upper bounds of the diffuse BSSRDF $R_d$, the geometry term $G$, and the Fresnel term $F_t$ in the triplet, respectively. The details for computing the upper bounds are shown in Appendix A.

### 4.3 BSSRDF-aware Cluster Selection

Once the triplet $(\mathbb{C}_j^L, \mathbb{C}_j^I, \mathbb{C}_j^S)$ with the maximum error bound is chosen, our method selects one cluster to subdivide the triplet. Previous scalable approaches [1], [4] basically use the diagonal lengths of the bounding boxes of the clusters as the criterion for selecting the cluster. This, however, does not work in our method, since we use uniform sampling over the surface of the translucent material for generating irradiance points, and irradiance points with large contributions to the shading points are rarely sampled until all the clusters are sufficiently subdivided.

Instead, we propose a BSSRDF-aware cluster selection method. As shown in Eq. (8), the variance of $\hat{\mathcal{I}}_j$ stems from the diffuse BSSRDF $R_d$ between $\mathbb{C}_j^I$ and $\mathbb{C}_j^S$, and $H$ between $\mathbb{C}_j^L$ and $\mathbb{C}_j^I$, since $A_j W_j I_j$ is constant in the triplet. In the product $R_d H$, the diffuse BSSRDF $R_d$ decreases exponentially with respect to the distance $r$ between the shading point and the irradiance point, and the geometry term $G$ decreases in inverse proportion to the square of the distance

$d$ between the VPL and the irradiance point. As such, our BSSRDF-aware cluster selection uses $R_d/d^2$ as the criterion. That is, our method selects the cluster with the maximum range of $R_d/d^2$ from $\mathbb{C}_j^L, \mathbb{C}_j^I$, and $\mathbb{C}_j^S$. Specifically, we define the range of $R_d/d^2$ for each cluster $c \in \{L, I, S\}$ when the other two clusters are fixed, $\Delta_c$ to be:

$$\Delta_c = \frac{R_d(r_{min})}{d_{min}^2} - \frac{R_d(r_{max})}{d_{max}^2}, \tag{10}$$

where $r_{max}$ and $r_{min}$ are the maximum and minimum distances between the irradiance points in $\mathbb{C}_j^I$ and the shading points in $\mathbb{C}_j^S$. $d_{max}$ and $d_{min}$ are the maximum and minimum distances between the VPLs in $\mathbb{C}_j^L$ and the irradiance points in $\mathbb{C}_j^I$, respectively.

To simplify the computation of the maximum and minimum distances between two clusters, our method uses the bounding sphere of each cluster in calculating $\Delta$ as shown in Fig. 3.

$$\Delta_L = \frac{R_d(r_{min})}{d_{min}^2} - \frac{R_d(r_c - r_I - r_S)}{(d_c - r_I + r_L)^2}, \tag{11}$$

$$\Delta_I = \frac{R_d(r_{min})}{d_{min}^2} - \frac{R_d(r_c - r_S + r_I)}{(d_c - r_L + r_I)^2}, \tag{12}$$

$$\Delta_S = \frac{R_d(r_{min})}{d_{min}^2} - \frac{R_d(r_c - r_I + r_S)}{(d_c - r_L - r_I)^2}, \tag{13}$$

where $d_c$ is the distance between the centers of the bounding spheres of $\mathbb{C}_i^L$ and $\mathbb{C}_i^I$, and $r_c$ is that of $\mathbb{C}_i^I$ and $\mathbb{C}_i^S$. $r_L$, $r_I$, and $r_S$ are the radii of the bounding spheres of $\mathbb{C}_j^L$, $\mathbb{C}_j^I$, and $\mathbb{C}_j^S$, respectively. Since $\Delta$ has a common upper bound $R_d(r_{min})/d_{min}^2$, our method only calculates each lower bound $R_d(r_{max})/d_{max}^2$ and selects the cluster with the minimum lower bound to be subdivided.

LC and SC are subdivided by traversing the corresponding binary trees. That is, the cluster corresponding to an inner node is replaced with two clusters corresponding to the child nodes in the binary tree. IC is subdivided in the same way by traversing the BVH for triangular meshes of the translucent material. When IC containing a single triangle is selected to subdivide, our method subdivides the triangle into two triangles by halving the longest edge and generates two ICs containing each subdivided triangle.

## 5 Results and Discussion

In this section, we compare our method with Multidimensional Lightcuts (MDLC) [1] that is combined with importance sampling of the BSSRDF [16] to handle subsurface rendering, and the error estimation method for Lightcuts [4] (we refer to this as *EELC*) in terms of the estimation accuracy and the computational efficiency. The estimation accuracy is measured by the ratio of pixels whose relative errors are less than $\epsilon$. To measure the estimation accuracy, our method uses only the pixels that cover the translucent materials. In all images, the relative error threshold $\epsilon$ was set to $2\%$, and the confidence level $\alpha$ for Eq. (9) was set to $95\%$. That is, the estimation is accurate when the ratio is close to $\alpha$ (i.e., $95\%$ in our experiments). The reference image is rendered by our method with the relative error threshold parameter $\epsilon = 0.1\%$. We also measured the root mean square error (RMSE) to assess the image quality.

### 5.1 Experimental setup

We tested our method on four scenes. The BSSRDF parameters are appropriately scaled from the measured parameters [6]. The first scene contains a single Buddha model with Skin material parameter. The second scene contains two Buddha models with Skim Milk and Marble parameters. The third scene has a human head model with texture. The BSSRDF parameters for the human head model are manually selected. In the *Salle de bain* scene, the walls, the washstand, and the bathtub are made of marble. The boundary surfaces of the translucent objects except for the human head model are specular materials. The image resolution for the Salle de bain scene is $720 \times 480$, and that for other scenes is $512 \times 512$. All images are rendered on a PC with an Intel Core i9-7980XE CPU.

At each pixel, $16$ rays are traced for supersampling and the intersection points are gathered as shading points. If a ray intersects a specular material, the intersection point is gathered as the shading point, and the reflected ray is recursively traced.

Both MDLC [1] and EELC [4] necessitate the user to specify the number of irradiance points. Unless otherwise stated, the number of irradiance points per shading point, $4096$, is chosen for EELC in all scenes so that the estimation accuracy of EELC comes close to that of our method. The same number is used in MDLC. For equal-time comparisons between EELC and our method, the numbers of irradiance points used in EELC are manually selected for each scene after several trial-and-error processes. The relative error threshold $\epsilon$ is set to $2\%$ for both MDLC and EELC, and the confidence level $\alpha$ is set to $95\%$ for EELC for comparisons.

### 5.2 Comparison to MDLC, EELC, and Path Tracing

Fig. 4 shows comparisons between MDLC (second column), EELC (third and fourth columns), our method (fifth column), and the reference (sixth column). As shown in Fig. 4 and Table 1, our estimation accuracy is very close to $\alpha$ in these scenes, indicating that our method can control the error accurately. The images rendered by our method are indistinguishable from the reference images as shown in Fig. 4. As shown in the second column (MDLC) in Fig. 4, the estimation accuracy with MDLC is low compared to our method and noticeable noise can be seen even for a large number of irradiance points, since MDLC fails to bound the total error from all the cluster triplets.

In an equal-time comparison to EELC (the third column in Fig. 4), the estimation accuracy of our method is up to 32.6 percentage points better than that of EELC in all scenes. Although EELC is capable of controlling the total error from the true value of the pixel intensity $\mathcal{I}$, the true value itself is not accurate when a small number of irradiance points is used. Therefore, noticeable noise can be seen especially in the Two Buddhas scene and the Head scene. The Salle de bain scene, which contains vast areas of translucent material (e.g., the walls), is a challenging scene for our method since sampling irradiance points with large contributions from vast areas requires a large number of partitions, resulting in prohibitively long computation times. Nevertheless, the estimation accuracy of our method ($94.8\%$) is better than

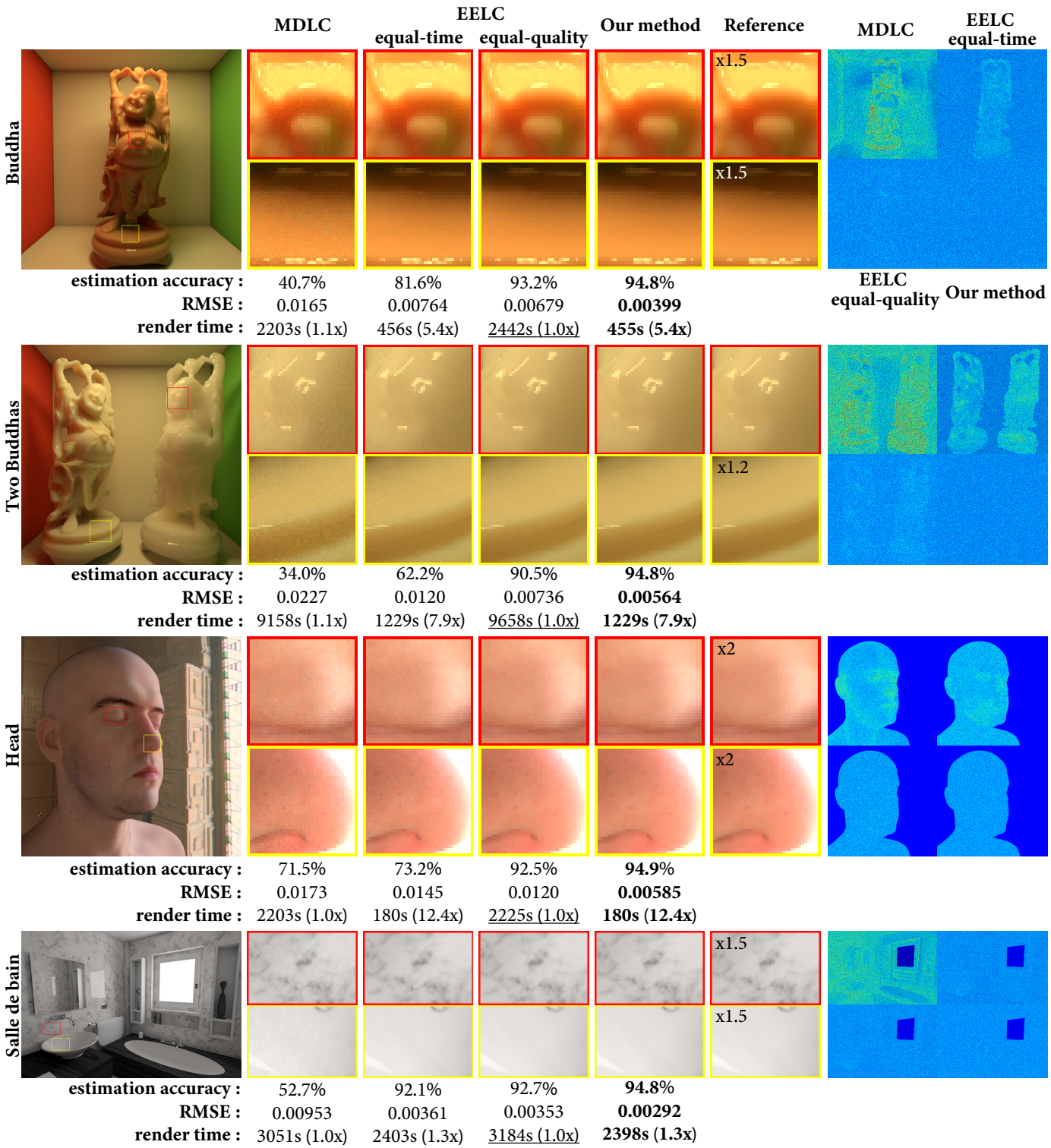


Fig. 4. Comparison between Multidimensional Lightcuts (MDLC) [1], Error Estimation for Lightcuts (EELC) [4], and our method. From left to right, the reference image (first), MDLC (second), EELC (third and fourth, where the render time and the estimation accuracy are set to be equal to those of our method by adjusting the number of the irradiance points in EELC), our method (fifth), and the reference (sixth). The rightmost images are the visualiation of the relative errors with false color. The percentage shown under each image indicates the estimation accuracy, which is the ratio of pixels whose relative errors are less than $\epsilon$. The RMSEs and the render times are also shown. To compare the render times, we set EELC of equal-quality (with $4096$ irradiance points per shading point) to be the baseline method (underlined). The numbers in the parenthesis show the speedups of each method compared to the baseline. The intensities of the insets are scaled for better clarity. Perceptible noise can be seen in images rendered by MDLC and EELC in equal-time, while our method does not provide perceptible noise thanks to our error estimation method. In equal-euqlity comparison, our method achieves up to 12.4x speedup.

TABLE 1
Statistics of the numbers of VPLs and the triangles used for four scenes, the render times, the estimation accuracy, and RMSEs for each method. Numbers with # indicate the number of the irradiance points per shading point used for the importance sampling in MDLC [1] and EELC [4]. The bold numbers indicate the best performance (highest estimation accuracy, lowest RMSEs and render times). In all scenes, our method achieves high estimation accuracy very close to the confidence level $\alpha = 95\%$ compared to MDLC and EELC in equal-time. Moreover, our method achieves a significant speedup compared to MDLC and EELC in equal-quality.

| Scene | Num. of VPL | Num. of tri. | Render times / estimation accuracy / RMSE ($\times 10^{-3}$) our method | | | MDLC (#4096) | | | EELC equal-time | | | EELC equal-quality (#4096) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Buddha | 0.998M | 1.09M | **455s** | **94.8%** | **3.99** | 2203s | 40.7% | 16.5 | 456s (#296) | 81.6% | 7.64 | 2442s | 93.2% | 6.79 |
| Two Buddhas | 1.037M | 2.17M | **1229s** | **94.8%** | **5.64** | 9158s | 34.0% | 22.7 | 1229s (#449) | 62.2% | 12.0 | 9658s | 90.5% | 7.36 |
| Head | 0.309M | 0.28M | **180s** | **94.9%** | **5.85** | 2203s | 71.5% | 17.3 | 180s (#323) | 73.2% | 14.5 | 2225s | 92.5% | 12.0 |
| Salle de bain | 1.018M | 1.22M | **2398s** | **94.8%** | **2.92** | 3051s | 52.7% | 9.53 | 2403s (#3029) | 92.1% | 3.61 | 3184s | 92.7% | 3.53 |

TABLE 2
Computation times for the importance sampling in EELC [4] with 4096 samples per shading point. Since the importance sampling of BSSRDF [27] involves costly ray-triangle intersections, the computation times dominate about 90% of the render times in all scenes.

| Scene | time for importance-sampling | render time | occupancy (%) |
|---|---|---|---|
| Buddha | 2155s | 2442s | 88.2% |
| Two Buddhas | 8985s | 9658s | 93.0% |
| Head | 2145s | 2225s | 96.4% |
| Salle de bain | 2897s | 3184s | 91.0% |

that of EELC (92.1%) in equal time, due to our BSSRDF-aware cluster selection method discussed in Sec. 5.3.

In an equal-quality comparison with EELC, 4096 irradiance points per shading point are required for EELC to achieve a high estimation accuracy, but it also incurs extensive computation times for importance-sampling. As shown in the fifth column of Fig. 4, our method achieves 1.3x to 12.4x speedups compared to EELC with equal-quality. Since importance sampling of the BSSRDF [16], [27] involves costly ray-triangle intersections, generating a large number of irradiance points to obtain accurate results can cause a bottleneck to the rendering. Table 2 shows the computation times for importance sampling of the irradiance points used in EELC (please note that the computational time for the importance sampling in EELC is same as that in MDLC since the importance sampling process is common in EELC and MDLC). As shown in Table 2, the computation times for the importance sampling account for about 90% in all four scenes. On the other hand, our adaptive sampling accounts for 9.17% (Buddha), 8.32% (Two Buddhas), 19.3% (Head), and 25.2% (Salle de bain) of each render time, respectively. Since our method samples the irradiance points uniformly on the triangles with a simple pdf, our method efficiently generates a large number of irradiance points.

The rightmost images in Fig. 4 visualize the relative errors in false color for each method (from left to right, top to bottom, MDLC, EELC in equal time, EELC with equal-quality, and our method). As shown in these images, our method consistently controls the relative errors to be less than the threshold $\epsilon$.

Table 1 lists the statistics for each scene (i.e., the numbers of VPLs and triangles), the estimation accuracy, the root mean square error (RMSE), and the computation time for each method. The number of irradiance points used for equal-time comparison with EELC in each scene is also listed. As shown in Table 1, our method yields both the best estimation accuracy and the smallest RMSEs compared to MDLC and EELC.

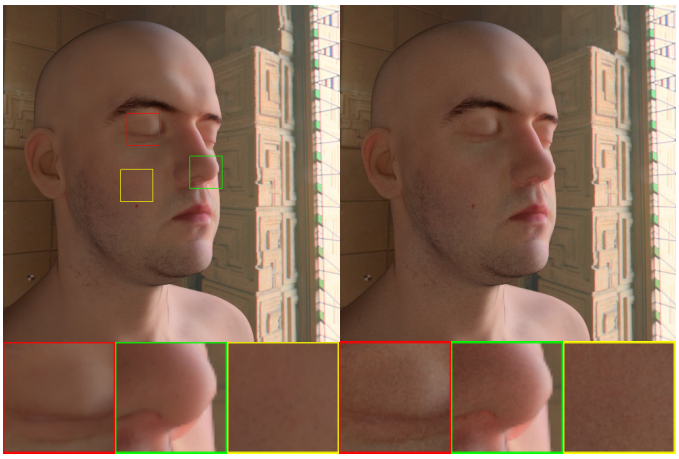

Fig. 5. Equal-time comparison between our method (left, 180s) and path tracing (right, 178s). As shown in the bottom insets, the noticeable noise can be seen in path tracing, while our method does not yield perceptible noise due to our error estimation.

Fig. 5 shows an equal-time comparison for the Head scene between our method and path tracing based on BSSRDF importance sampling [27]. Although path tracing is good at handling such a scene (i.e., diffuse BSSRDF material and a large light source represented by an environment map), the rendering results by path tracing produce noticeable noise as shown in the insets at the bottom Fig. 5. On the other hand, our method does not yield noticeable noise since our method can accurately control the error.

Fig. 6 shows convergence plots of the RMSE for MDLC, EELC, and our method by changing the relative error threshold parameter $\epsilon$. In Fig. 6, we include the convergence plots for MDLC and EELC using uniform sampling of the irradiance points, which are explicitly referred to as MDLC(US) and EELC(US) to distinguish them from those using importance sampling. Both MDLC(US) and EELC(US) with 4096 irradiance points per shading point are fast but provide large errors. Increasing the number of irradiance points per shading point up to 65536 (i.e., at least $1M$ irradiance points per pixel) does not perform well compared to MDLC and EELC. For MDLC and EELC, since the computation time for importance sampling dominates the render time, the change in the relative error threshold parameter $\epsilon$ has little impact on the render time when $\epsilon \geq 0.5\%$. As shown in Fig. 6, the RMSEs using our method are smaller

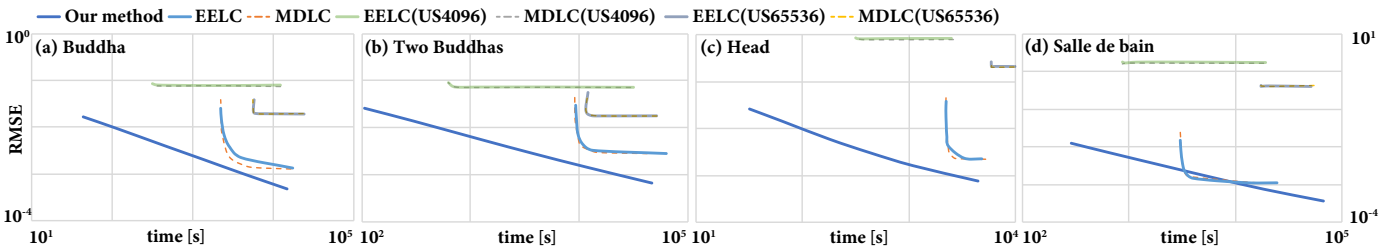


Fig. 6. Convergence plots of RMSEs for our method (dark blue solid line), EELC with importance sampling (sky blue solid line) and uniform sampling (light green/light purple solid lines), and MDLC with importance sampling (dark orange dashed line) and uniform sampling (light gray/light orange dashed lines) on the log-log scale. RMSEs and the render times are measured by changing the relative error threshold $\epsilon$. The numbers after "US" (i.e., $4096$ and $65536$) indicate the number of the irradiance points per shading point used in uniform sampling. Both MDLC(US) and EELC(US) do not perform well compared to MDLC, EELC, and our method. While the error reduction rates in EELC and MDLC begin to deteriorate when tight error thresholds ($\epsilon < 0.5\%$ for EELC and $\epsilon < 0.05\%$ for MDLC) are used, our method reduces RMSEs steadily thanks to our adaptive sampling.

than those for EELC and MDLC in all scenes except for the Salle de bain scene. In the Salle de bain scene, RMSEs of our method with $\epsilon = 1\%$, EELC with $\epsilon = 0.5\%$, and MDLC with $\epsilon = 0.05\%$ are similar (0.001476, 0.00144, and 0.001582, respectively), but the render times for EELC with $\epsilon = 0.5\%$ (4007s) and MDLC with $\epsilon = 0.05\%$ (3659s) are smaller than that of our emthod with $\epsilon = 1\%$ (6622s). However, the error reduction rates in EELC and MDLC drop abruptly when $\epsilon < 0.5\%$ for EELC and $\epsilon < 0.05\%$ for MDLC as shown in Fig. 6. On the other hand, our method yields stable error reduction and better convergence compared to EELC. We also discuss the similar behavior of the estimation accuracy in Sec. 5.4.3.

### 5.3 Cluster Selection Comparison

We investigate the effect of our BSSRDF-aware cluster selection method on the rendering performance. We compare our selection method with the previous method that selects the cluster with the maximum extent (i.e., with the largest diagonal length of the bounding box) from three clusters of a triplet. The previous cluster selection method is incorporated into our uniform sampling of the triangles and the irradiance points. We refer to the previous cluster selection method used in MDLC and EELC as *Max-Extent*. Fig. 7 visualizes the number of triplets $J$ used to control the relative error to less than the threshold $\epsilon = 2\%$. The top row visualizes the number of triplets partitioned by using our BSSRDF-aware cluster selection method. In all scenes, the shadow regions (e.g., under the jaw and around the foot of the Buddha model, the corner of the walls, and the shelf under the window in the Salle de bain scene) have many triplets that satisfy the termination criterion $\Delta\hat{\mathcal{I}} < \epsilon\hat{\mathcal{I}}$, since the tolerance $\epsilon\hat{\mathcal{I}}$ is small in the shadow regions. This property is common to MDLC and EELC, and we leave the solution to this for future work. The bottom row in Fig. 7 visualizes the number of triplets partitioned by using Max-Extent. Compared to the top row in Fig. 7, the number of triplets in our method is much smaller (1.79x to 7.65x) than that for Max-Extent to satisfy the termination criterion $\Delta\hat{\mathcal{I}} < \epsilon\hat{\mathcal{I}}$.

Table 3 shows comparisons between the render times for our method and Max-Extent. The estimation accuracy and RMSEs for Max-Extent are also shown in Table 3. Although Max-Extent can achieve high estimation accuracy and RMSEs similar to ours, the render times also increase by up to 8.4 times. This indicates that Max-Extent requires excessive subdivisions of the triplets to generate irradiance points with large contributions. Although our method samples the triangles and the irradiance points uniformly, our BSSRDF-aware cluster selection method can efficiently sample irradiance points with large contributions. The numbers in parentheses show the improvements in render times and the numbers of the triplets for our method. Since the computation cost of our method is linear with respect to the number of triplets $J$, this number directly affects the render time as shown in Table 3. As expected, the improvements in the numbers of triplets using our method are similar to those for the render times (both numbers are shown in the parentheses in Table 3).

TABLE 3
Statistics of the render times, the average numbers of the triplets, the estimation accuracy, and RMSEs for the previous cluster selection method (Max-Extent). The numbers in the parenthesis show the ratio of the render times and the average numbers of the triplets of Max-Extent with respect to our method.

| Scene | Max-Extent | | | | | |
|---|---|---|---|---|---|---|
| | render time | | #triplets | | accuracy | RMSE |
| Buddha | 3325s | (7.31x) | 141k | (7.65x) | 94.8% | 0.00314 |
| Two Buddhas | 4760s | (3.87x) | 93.1k | (3.84x) | 94.9% | 0.00510 |
| Head | 330s | (1.83x) | 15.7k | (1.79x) | 94.8% | 0.00519 |
| Salle de bain | 20175s | (8.41x) | 359k | (6.67x) | 94.9% | 0.00294 |

### 5.4 Discussion

#### 5.4.1 Application to Path Tracing

Our method has the potential to be integrated into a Monte Carlo path tracing framework. While we partition the integral domain into the triplets for the error estimation, partitioning the integral domain (i.e., stratified sampling) can be used for sampling light paths. Let us explain how to sample a light path $\bar{x} = x_0x_1x_2x_3$ with length three by using our method, where $x_0$ and $x_3$ are on the camera and a light source (not restricted to VPL), and $x_1$ (corresponding to the shading point) and $x_2$ (corresponding to the irradiance point) are on the surfaces of a translucent material. Light paths with length $k \geq 3$ can be sampled by continuing to trace the path from $x_2$ in the similar way as path tracing.

We first sample a shading point $x_1^1$, an irradiance point $x_2^1$, and a point on the light source $x_3^1$ from the root triplet $(\mathbb{C}_1^L, \mathbb{C}_1^I, \mathbb{C}_1^S)$. This corresponds to sampling a light path

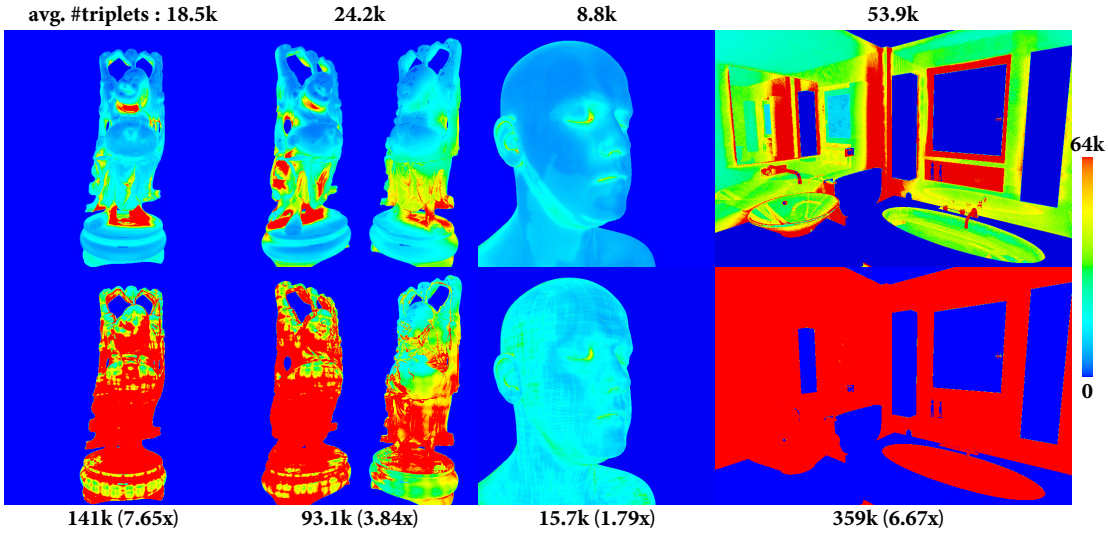


Fig. 7. Visualization of the number of the triplets in false color using our BSSRDF-aware cluster selection method (top) and the previous selection method, Max-Extent, (bottom). The average numbers of the triplets for both selection methods are listed. The numbers in the parenthesis show the improvement of our method with respect to the previous method.

$\bar{x}_1 = x_0 x_1^1 x_2^1 x_3^1$ as shown in Fig. 8 (left). Next, we select a cluster to be subdivided using our BSSRDF-aware cluster selection method. If, for instance, IC $\mathbb{C}_1^L$ is selected as shown in Fig. 8 (right), and the root triplet is subdivided into two triplets $(\mathbb{C}_1^L, \mathbb{C}_2^I, \mathbb{C}_1^S)$ and $(\mathbb{C}_1^L, \mathbb{C}_3^I, \mathbb{C}_1^S)$, a new light path $\bar{x}_2$ is sampled by sampling $x_1^2$, $x_2^2$, and $x_3^2$ from the triplet $(\mathbb{C}_1^L, \mathbb{C}_3^I, \mathbb{C}_1^S)$ as shown in Fig. 8 (right). The light path $\bar{x}_1$ sampled in the previous step can be reused by updating the sampling pdf for the irradiance point $x_2^1$ since $x_2^1$ is sampled from $\mathbb{C}_2^I$ instead of $\mathbb{C}_1^I$. By repeating these processes, our method can sample the light paths in the path tracing framework.

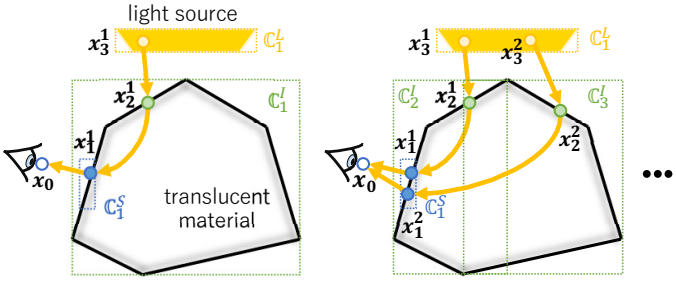


Fig. 8. Our method can be adapted to sample light paths with length three in a path tracing framework. $x_0$ is on the camera, and $x_3^i$ is the $i$-th sample on the light source. $x_1^i$ corresponding to the shading point is sampled from SC $\mathbb{C}^S$ and $x_2^i$ corresponding to the irradiance point is sampled from IC $\mathbb{C}^I$. Left: a light path $\bar{x}_1 = x_0 x_1^1 x_2^1 x_3^1$ is sampled from the triplet $(\mathbb{C}_1^L, \mathbb{C}_1^I, \mathbb{C}_1^S)$. Right: IC $\mathbb{C}_1^L$ is partitioned into $\mathbb{C}_2^L$ and $\mathbb{C}_3^L$, and another light path $\bar{x}_2 = x_0 x_1^2 x_2^2 x_3^2$ is sampled. By repeating this process, multiple light paths are sampled.

### 5.4.2 Scalability

We experimented on the scalability of our method with respect to the VPL count. Fig. 9 shows the relationship between the numbers of VPLs and the render times by doubling (or halving) the numbers of VPLs listed in Table 1. By increasing the number of VPLs eight times, the increases in render time for each scene are only 13% (Buddha), 17% (Two Buddhas), 15% (Head), and 21% (Salle de bain), respectively. As shown in Fig. 9, our method scales well with respect to the VPL count. In contrast, our method scales linearly with respect to the number of irradiance points, while MDLC and EELC scales sublinearly due to the hierarchical structure of the (pre-sampled) irradiance points. However, as shown in Table 2, a computational bottleneck resides in importance sampling of the irradiance points, and the cost for importance sampling scales linearly with the number of irradiance points.

### 5.4.3 Robustness to various relative error thresholds

Fig. 10 shows graphs of the estimation accuracy with different relative threshold parameter ($\epsilon$) values. We compared the estimation accuracy between our method and EELC (with 4096 irradiance points per shading point) by changing $\epsilon$ from 10% to 0.5%. As shown in Fig. 10, the estimation accuracy of EELC for $\epsilon \geq 2\%$ is close to 95%, while this falls sharply to 70% in all the scenes when $\epsilon = 0.5\%$ is specified. This is because the maximum number of irradiance points is fixed for EELC, and additional irradiance points are required to achieve a tight error threshold. On the other hand, the estimation accuracy of our method is consistently very close to 95% for different $\epsilon$, indicating that our adaptive sampling of the irradiance points is robust in achieving high estimation accuracy against various relative error threshold parameter values.

### 5.4.4 Mesh quality

Since our method clusters the triangular mesh, we investigated the effect of the quality of the triangular mesh on the rendering performance in the Salle de bain scene. The left wall in the Salle de bain scene consists of only two triangles, which indicates that our method works well for low-resolution triangular meshes. We subdivided the two triangles of each wall 128 times, then we applied a finer triangular mesh to our error estimation method. The render time was $2430s$, and we could not see any significant

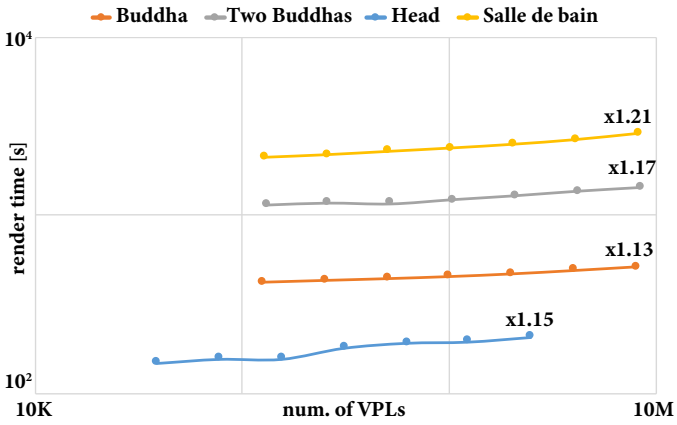


Fig. 9. Plots of the render times and the number of VPLs on the log-log scale show the scalability of our method. The number above each plot shows the increase in the render time when the number of VPLs is increased by eight times. This indicates that our method scales well to the VPL count.

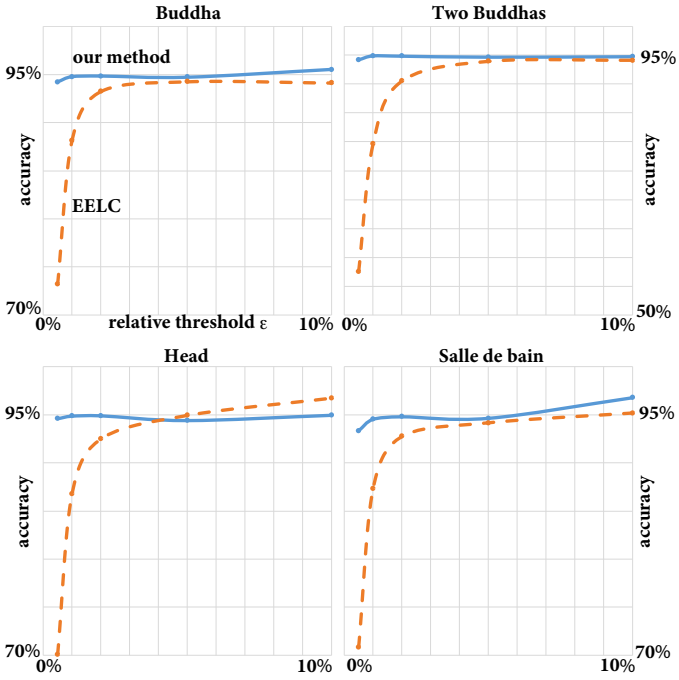


Fig. 10. Comparison of the estimation accuracy betwen our method (solid line) and EELC (dashed line) for various relative error threshold ($\epsilon$) values. We measured the estimation accuracy for $\epsilon = 10\%, 5\%, 2\%, 1\%,$ and $0.5\%$ with the same confidence level $\alpha = 95\%$. The estimation accuracy for EELC decreases for tight error thresholds (e.g., it falls down to around $70\%$ in all the scenes when $\epsilon = 0.5\%$), while that of our method is consistently close to $95\%$.

difference to that ($2398s$) obtained with the original low triangular mesh. We also measured the render times for the Buddha scene by subdividing the triangular mesh of the Buddha model (1.09M triangles). The render time increases by $13.0\%$ and $22.8\%$ when the original triangular mesh is refined to 2.92M ($2.68\times$) and 3.99M ($3.66\times$) triangles, respectively. The computational overhead is sublinear with respect to the number of triangles.

### 5.4.5 *Image resolution*

Finally, we applied our method to higher resolution images. We rendered the Head scene with a resolution of $1024^2$ with supersampling (4 rays per pixel). The estimation accuracy and RMSE are $94.9\%$, and $0.0062$, respectively, which are similar to those with a resolution of $512^2$ (see Table 1). The render time for the Head scene for $1024^2$ is 600s. This indicates that our error estimation framework works well regardless of the image resolution.

## 6 Conclusions

We have proposed an adaptive sampling method of the irradiance points for subsurface rendering with the diffuse BSSRDF model so that the relative error is less than the user specified threshold. We proposed a BSSRDF-aware cluster selection method to efficiently partition the triplet. We showed that our method can eliminate the need for specifying the number of the irradiance points, and our method can render images indistinguishable from the reference solution without a tedious trial-and-error process and re-renderings. Besides, thanks to the use of simple uniform sampling combined with the BSSRDF-aware cluster selection, our method achieved significant speedups compared to the previous methods that require costly importance sampling.

Although our method lifts the limitation of the previous scalable subsurface rendering methods, our method still has the following limitations. Our method necessitates specifying the numbers of shading points. A small number of shading points leads to aliasing and artifacts in the rendered images. However, in our method, we can identify that the cause of such artifacts comes from the lack of shading points, efforts to tweak parameters are smaller than those in previous methods. Since our method is based on the many-light rendering framework, our method inherits the disadvantages of many-light rendering, such as the singularities due to the geometry term. However, since our method can handle a large number of VPLs efficiently as shown in Table. 1 and Fig. 9, those singularities can be mitigated.

In future work, we would like to investigate whether more sophisticated BSSRDF models can be applied to our error estimation framework. Moreover, we would like to extend our method to heterogeneous translucent materials.

## References

[1] B. Walter, A. Arbree, K. Bala, and D. P. Greenberg, "Multidimensional lightcuts," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1081–1088, 2006.

[2] A. Arbree, B. Walter, and K. Bala, "Single-pass scalable subsurface rendering with lightcuts," *Computer Graphics Forum*, vol. 27, no. 2, pp. 507–516, 2008.

[3] Y. T. Wu, T. M. Li, Y. H. Lin, and Y. Y. Chuang, "Dual-matrix sampling for scalable translucent material rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 3, pp. 363–374, 2015.

[4] H. Sakai, K. Nabata, S. Yasuaki, and K. Iwasaki, "A method for estimating the errors in many-light rendering with supersampling," *Computational Visual Media*, vol. 5, pp. 151–160, 2019.

[5] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. AMS, 2003.

[6] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Proc. of SIGGRAPH'01*, 2001, pp. 511–518.

[7] C. Donner and H. W. Jensen, "Light diffusion in multi-layered translucent materials," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1032–1039, 2005.
[8] E. D'Eon and G. Irving, "A quantized-diffusion model for rendering translucent materials," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 56:1–56:14, 2011.
[9] R. Habel, P. H. Christensen, and W. Jarosz, "Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering," *Computer Graphics Forum*, vol. 32, no. 4, pp. 27–37, 2013.
[10] J. R. Frisvad, T. Hachisuka, and T. K. Kjeldsen, "Directional dipole model for subsurface scattering," *ACM Transactions on Graphics*, vol. 34, no. 1, pp. 5:1–5:12, 2014.
[11] R. Frederickx and Ph. Dutré, "A forward scattering dipole model from a functional integral approximation," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 109:1–109:13, 2017.
[12] A. Keller, "Instant radiosity," in *Proc. of SIGGRAPH '97*, 1997, pp. 49–56.
[13] C. Dachsbacher, J. Krivanek, M. Hasan, A. Arbree, B. Walter, and J. Novak, "Scalable realistic rendering with many-light methods," *Computer Graphics Forum*, vol. 33, no. 1, pp. 88–104, 2014.
[14] J. Křivánek, I. Georgiev, A. Kaplanyan, and J. Canad, "Recent advances in light transport simulation: Theory and practice," in *ACM SIGGRAPH 2013 Courses*, 2013.
[15] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg, "Lightcuts: A scalable approach to illumination," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1098–1107, 2005.
[16] B. Walter, P. Khungurn, and K. Bala, "Bidirectional lightcuts," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 59:1–59:11, 2012.
[17] N. Bus, N. H. Mustafa, and V. Biri, "Illuminationcut," *Computer Graphics Forum*, vol. 34, no. 2, pp. 561–573, 2015.
[18] M. Hašan, F. Pellacini, and K. Bala, "Matrix row-column sampling for the many-light problem," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 26:1–26:10, 2007.
[19] J. Ou and F. Pellacini, "Lightslice: Matrix slice sampling for the many-lights problem," *ACM Transactions on Graphics*, vol. 30, no. 6, pp. 179:1–179:8, 2011.
[20] Y.-T. Wu and Y.-Y. Chuang, "Visibilitycluster: Average directional visibility for many-light rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 9, pp. 1566–1578, 2013.
[21] Y. Huo, R. Wang, S. Jin, X. Liu, and H. Bao, "A matrix sampling-and-recovery approach for many-lights rendering," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 210:1–210:12, 2015.
[22] K. Nabata, K. Iwasaki, Y. Dobashi, and T. Nishita, "An error estimation framework for many-light rendering," *Computer Graphics Forum*, vol. 35, no. 7, pp. 431–439, 2016.
[23] C. R. A. Chaitanya, L. Belcour, T. Hachisuka, S. Premoze, J. Pantaleoni, and D. Nowrouzezahrai, "Matrix bidirectional path tracing," in *Proceedings of EGSR (Experimental Ideas & Implementations)*, 2018.
[24] Y. Tokuyoshi and T. Harada, "Hierarchical russian roulette for vertex connections," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 36:1–36:12, 2019.
[25] H. W. Jensen and J. Buhler, "A rapid hierarchical rendering technique for translucent materials," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 576–581, 2002.
[26] D. Milaenen, L. Belcour, J.-P. Guertin, T. Hachisuka, and D. Nowrouzezahrai, "A frequency analysis and dual hierarchy for efficient rendering of subsurface scattering," in *Proceedings of Graphics Interface 2019*, 2019.
[27] A. King, C. Kulla, A. Conty, and M. Fajardo, "BSSRDF importance sampling," in *ACM SIGGRAPH 2013 Talks*, 2013, pp. 48:1–48:1.
[28] Y. Tokuyoshi and T. Harada, "Stochastic light culling for VPLs on GGX microsurfaces," *Computer Graphics Forum*, vol. 36, no. 4, pp. 55–63, 2017.

**Kosuke Nabata** received his B.S., M.S., and Ph.D degrees from Wakayama University in 2013, 2015, and 2019, respectively.

**Kei Iwasaki** received his B.S., M.S., and Ph.D. degrees from the University of Tokyo, in 1999, 2001, and 2004, respectively. He is currently an associate professor at Wakayama University.