Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

An Error Estimation Framework for Many-Light Rendering

K. Nabata[1], K. Iwasaki[1,3], Y. Dobashi[2,3], and T. Nishita[3,4]

[1]Wakayama University, Japan
[2]Hokkaido University, Japan
[3]UEI Research, Japan
[4]Hiroshima Shudo University, Japan

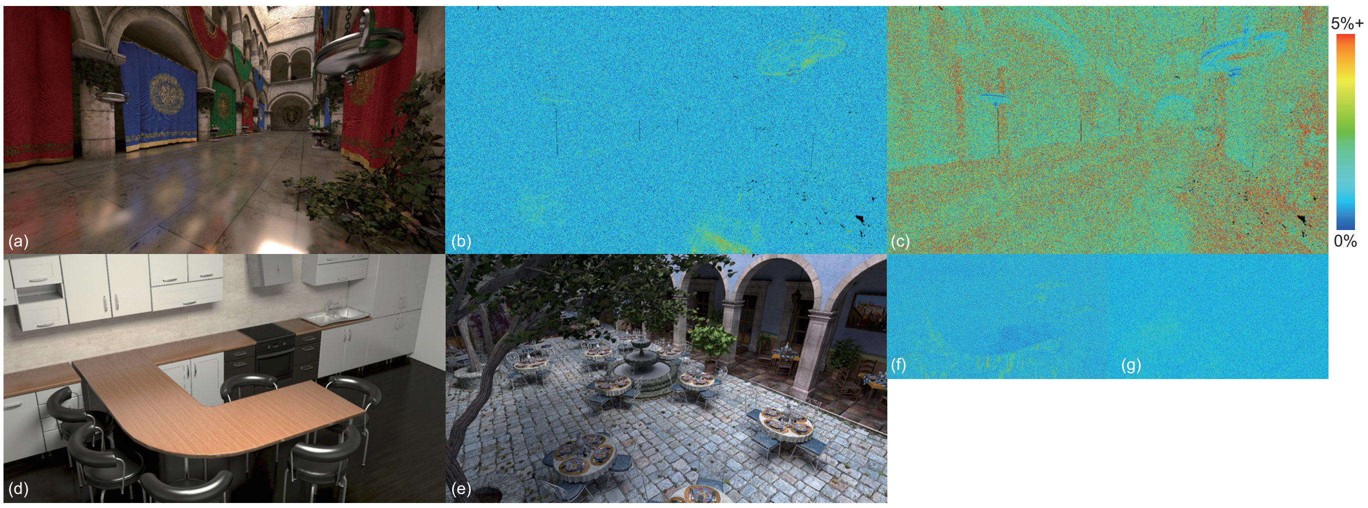

Figure 1: *Rendering results of (a) our method, relative error images of (b) our method and (c) Lightcuts [WFA*05] in Sponza scene. Relative error threshold $\varepsilon = 2\%$ and confidence level $\alpha = 95\%$ are specified. 92.96% pixels satisfy that the relative error is within 2% in our method, while only 43.67% pixels satisfy the condition in Lightcuts. In (d) Kitchen scene and (e) San Miguel scene, Cook-Torrance BRDFs and Ashikhmin-Shirely BRDFs are used, which cannot be used in Lightcuts. (f) and (g) show relative error images of (d) and (e), respectively.*

Abstract

The popularity of many-light rendering, which converts complex global illumination computations into a simple sum of the illumination from virtual point lights (VPLs), for predictive rendering has increased in recent years. A huge number of VPLs are usually required for predictive rendering at the cost of extensive computational time. While previous methods can achieve significant speedup by clustering VPLs, none of these previous methods can estimate the total errors due to clustering. This drawback imposes on users tedious trial and error processes to obtain rendered images with reliable accuracy. In this paper, we propose an error estimation framework for many-light rendering. Our method transforms VPL clustering into stratified sampling combined with confidence intervals, which enables the user to estimate the error due to clustering without the costly computing required to sum the illumination from all the VPLs. Our estimation framework is capable of handling arbitrary BRDFs and is accelerated by using visibility caching, both of which make our method more practical. The experimental results demonstrate that our method can estimate the error much more accurately than the previous clustering method.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Efficient rendering methods to predict visual appearances are necessary for many applications including design software (e.g. Autodesk 360 [AUT13]) and in the film industry. For such applications, designers and manufacturers need to rely on rendered images to design, create, and manipulate products, and thus the rendered images are expected to predict the visual appearance faithfully.

In recent years, many-light rendering [DKH*13] has attracted attention due to its simplicity, efficiency, and wide applicability. Many-light rendering simplifies complex global illumination computations by using a large number of virtual point lights (VPLs). The incident illumination onto points to be shaded (we refer to these points as shading points) is calculated by using VPLs. The outgoing radiance from a shading point is the sum of the contributions from all VPLs. For many-light rendering, the accuracy of the rendering increases as the number of VPLs increase. Therefore, an exact solution that sums up the contributions from all the VPLs is computationally expensive. To address this problem, clustering methods [WFA*05, WABG06, WKB12, HPB07, OP11, BMB15] for VPLs have been proposed with which efficient, scalable rendering with a huge number of VPLs can be provided. These methods cluster similar VPLs and the representative VPL for each cluster is used to calculate the outgoing radiance at each shading point. Unfortunately, none of these methods proposes an error estimation framework for many-light rendering, which results in tedious trial and error processes to produce images whose errors from the reference solution are guaranteed to be less than user-specified thresholds for predictive rendering.

We propose an error estimation framework for many-light rendering by using confidence intervals. Our method builds upon scalable clustering approaches (e.g. Lightcuts [WFA*05]) to estimate the error of the outgoing radiance directly, while the previous methods [WFA*05, BMB15] can only estimate the maximum error of each cluster instead of that of the outgoing radiance. By specifying the relative error threshold and a confidence interval parameter, our method automatically partitions a huge number of VPLs into clusters so that the outgoing radiance estimated by sampling two representative VPLs from each cluster is within the stochastic bound of the true value. Contrary to the previous methods [WFA*05, BMB15] that limit BRDFs whose upper bounds within the VPL cluster can be computed cheaply and tightly, our method can handle arbitrary BRDFs whose upper bounds are estimated by using BRDF importance sampling. Furthermore, our method introduces a visibility caching method to accelerate VPL rendering. Unlike previous caching methods [GKPS12, YNI*15] that record the visibilities between sparsely sampled points (called *cache points*) and all the VPLs, our method records the average visibilities between the cache points and the clusters of VPLs, whose number is much smaller than that of the VPLs, which reduces both the computational time and memory footprint required for the visibilities at the cache points.

The contributions of our method are as follows.

- An error estimation framework for VPLs is proposed. Our framework can be easily incorporated into Lightcuts [WFA*05].
- Our method removes the limitation of BRDFs applicable to Lightcuts, and thus widens the applicability of VPL rendering further.
- A visibility caching method is introduced to accelerate rendering without noticeable decrease in the error estimation accuracy.

2. Previous Work

2.1. Many-light Rendering Methods

Recent work on many-light methods [DKH*13, KGKC13] has revealed that global illumination effects can be well approximated by using many virtual light sources. Keller introduced the instant radiosity method, which calculates the indirect illumination by using virtual point lights (VPLs) [Kel97]. Following this seminal work, a number of many-light methods that can handle highly glossy materials [HKWB09, DKH*10, SHD15] and can achieve interactive rendering [RGK*08, Tok15], have been proposed.

Clustering VPLs: Walter et al. proposed a scalable solution to many-light methods using a hierarchical representation of VPLs, called *Lightcuts* [WFA*05, WABG06, WKB12]. Hasan et al. represent the many-light method as a large matrix, and explore the matrix structure by using row-column sampling [HPB07]. Ou and Pellacini [OP11] cluster the shading points into groups called slices and perform row-column sampling for each slice. Wang et al. [WHY*13] proposed an out-of-core many-light rendering method. Huo et al. [HWJ*15] proposed a matrix sampling-and-recovery approach for many-light rendering. Bus et al. [BMB15] cluster the product-space of all shading point-VPL pairs and achieve speedup compared to Lightcuts.

Sampling VPLs: Wang and Akerlund presented a BRDF importance sampling method for VPLs [WA09]. Georgiev et al. proposed an importance sampling method for VPLs by recording the contributions of VPLs at each cache point [GKPS12]. Wu and Chuang [WC13] proposed the VisibilityCluster, which approximates the visibilities between each cluster of VPLs and those of the shading points by estimating the average visibility. Yoshida et al. [YNI*15] improved the importance caching [GKPS12] by adaptively inserting cache points.

Although many methods have been proposed for many-light rendering, an error estimation framework has not been proposed. Lightcuts attempts to control the error due to clustering by estimating the upper bound error of each cluster, but cannot control the total error from all clusters. This results in tedious trial and error processes to obtain rendered images whose errors from the reference solution are bounded.

Visibility Caching: Clarberg et al. proposed a visibility caching method for direct illumination by storing 2D visibility maps at each cache point [CAM08]. Popov et al. quantize the visibility functions between two points by those between two clusters [PGSD13]. Ulbrich et al. presented a progressive visibility caching approach using shadow maps [UNRD13]. Although these methods can accelerate rendering significantly by using visibility caching, they introduce errors due to quantization [PGSD13] and discretization [CAM08, UNRD13], which makes the error estimation more complex.

2.2. Rendering with Confidence Interval

Purgathopher [Pur87] introduced the confidence intervals to control the number of samples for distributed ray tracing. Tamstorf and Jensen [TJ97] proposed an adaptive sampling method for path tracing using confidence intervals. Rigau et al. [RFS03] presented refinement criteria using convex functions. Hachisuka et al. [HJJ10] proposed an error estimation framework for progressive photon mapping. Dammertz et al. [DHKL10] introduced a hierarchical automatic stopping criterion for Monte-Carlo methods. Moon et al. [MJL*13] used confidence intervals to detect homogeneous pixels for image denoising. These methods, however, do not estimate the errors of many-light rendering.

2.3. Preliminary of Lightcuts

Since our method builds upon the scalable clustering approaches, we briefly review Lightcuts. In many-light rendering, the outgoing radiance $L(x,x_v)$ at the shading point x towards the viewpoint x_v is calculated from the following equation.

$$L(x,x_v) = \sum_{i=1}^{N_{vpl}} I(y_i) f(x_v,x,y_i) G(x,y_i) V(x,y_i), \tag{1}$$

where N_{vpl} is the number of VPLs, $I(y_i)$ is the intensity of the i-th VPL y_i, $f(x_v,x,y_i)$ is the BRDF, $G(x,y_i)$ is the geometry term, and $V(x,y_i)$ is the visibility term that returns 1 if x and y_i are mutually visible and returns 0 otherwise. To render realistic images, a large number of VPLs are usually required. In this case, summing the VPL contribution $I(y_i)f(x_v,x,y_i)G(x,y_i)V(x,y_i)$ for all VPLs is computationally expensive. Rather than summing all VPL contributions, Lightcuts partitions similar VPLs into N clusters, and the outgoing radiance $L_{\mathbb{C}_i}(x,x_v)$ illuminated by VPLs in cluster $\mathbb{C}_i$ is calculated from

$$L_{\mathbb{C}_i}(x,x_v) = \sum_{y \in \mathbb{C}_i} I(y) f(x_v,x,y) G(x,y) V(x,y). \tag{2}$$

Instead of summing all the VPLs in cluster $\mathbb{C}_i$, $L_{\mathbb{C}_i}$ is estimated by sampling N_S VPLs with probability p using the following equation.

$$\hat{L}_{\mathbb{C}_i}(x,x_v) = \frac{1}{N_S} \sum_{s=1}^{N_S} \frac{I(y_s) f(x_v,x,y_s) G(x,y_s) V(x,y_s)}{p(y_s)}. \tag{3}$$

Lightcuts samples the VPLs based on the probability proportional to the intensity (i.e. $p(y_s) = I(y_s)/\sum_{y\in\mathbb{C}_i} I(y)$) as in the following equation.

$$\begin{aligned}\hat{L}_{\mathbb{C}_i}(x,x_v) &= \frac{1}{N_S} \sum_{s=1}^{N_S} \frac{I(y_s) f(x_v,x,y_s) G(x,y_s) V(x,y_s)}{I(y_s)/\sum_{y\in\mathbb{C}_i} I(y)} \\ &= \frac{I_i}{N_S} \sum_{s=1}^{N_S} f(x_v,x,y_s) G(x,y_s) V(x,y_s),\end{aligned} \tag{4}$$

where $I_i = \sum_{y\in\mathbb{C}_i} I(y)$ is the sum of intensity, and Lightcuts sampled one representative VPL (i.e. $N_S = 1$ is used). The standard variation σ_i for $\mathbb{C}_i$ is calculated using the following equation (arguments are omitted for brevity and $p = I/I_i$ is substituted).

$$\sigma_i = \sqrt{\mathrm{Var}[IfGV/p]} = I_i \sqrt{\mathrm{Var}[fGV]}. \tag{5}$$

Lightcuts clusters VPLs for each shading point and represents VPLs and clusters with a binary tree, referred to as *light tree*. In the light tree, each leaf node corresponds to each VPL and each inner node corresponds to the cluster that contains all the VPLs of the descendant nodes. A set of clusters is represented by *cut*, where every path from the root node to each leaf node contains one node of the cut. From the root node, cut $\mathbb{C}$ is calculated by subdividing each cluster until each cluster $\mathbb{C}_i$ in $\mathbb{C}$ satisfies $E_{\mathbb{C}_i} < \varepsilon \hat{L}(x,x_v)$, where ε is a threshold specified by the user, and $E_{\mathbb{C}_i}$ is the upper bound of the clustering error calculated from the following equation.

$$E_{\mathbb{C}_i} = I_i f_{ub}(x_v,x,\mathbb{C}_i) G_{ub}(x,\mathbb{C}_i) V_{ub}, \tag{6}$$

where f_{ub}, G_{ub}, and V_{ub} represent the upper bound of each function within the cluster $\mathbb{C}_i$, respectively. The upper bound of visibility V_{ub} is set to 1. The estimated outgoing radiance $\hat{L}(x,x_v)$ is calculated by $\hat{L}(x,x_v) = \sum_{i=1}^{N} \hat{L}_{\mathbb{C}_i}(x,x_v)$.

Although Lightcuts can render realistic images efficiently, it has some limitations as follows. Firstly, Lightcuts do not directly estimate the total error $||L(x,x_v) - \hat{L}(x,x_v)||$ (i.e. the sum of errors of each cluster $\mathbb{C}_i$). Although it would be possible to estimate the total error by improving Lightcuts to sum the upper bound of clustering errors as $\sum_{i=1}^{N} E_{\mathbb{C}_i} < \varepsilon \hat{L}$, this overestimates the total error and may lead to a numerous number of clusters and expensive computational time. Secondly, applicable materials in Lightcuts are limited to Lambertian, Blinn-Phong, and isotropic Ward BRDFs only, since the calculation method of the upper bound f_{ub} is currently developed only for these three BRDF models.

Algorithm 1 Our Algorithm (**Input** : ε and α, **Output** : $\hat{L}$). Procedures of our error estimation framework are highlighted with underlines, and the rest of procedures are identical to Lightcuts.

```
1: Priority Queue : $Q \leftarrow \mathbb{C}_1$, Cut : $\mathbb{C} \leftarrow \mathbb{C}_1$
2: $\underline{\text{Estimate}}$ $\hat{L}$, $\underline{\Delta L, \text{ and } \sigma_1}$
3: while $\underline{\Delta L > \varepsilon\hat{L}} \vee \sqrt{2}\sigma_i > \varepsilon\hat{L}$ $(\forall \mathbb{C}_i \in \mathbb{C})$ do
4:     $\mathbb{C}_k \leftarrow Q.pop()$
5:     Replace $\mathbb{C}_k$ in $\mathbb{C}$ with two children $\mathbb{C}_L$ and $\mathbb{C}_R$
6:     $\underline{\text{Estimate}}$ $\hat{L}_{\mathbb{C}_L}$, $\hat{L}_{\mathbb{C}_R}$, $\underline{\sigma_L, \text{ and } \sigma_R}$
7:     Update $\hat{L}$ and $\underline{\Delta L}$
8:     $\underline{Q \leftarrow \mathbb{C}_L, \mathbb{C}_R}$
9: end while
```

3. Error Estimation Framework for VPLs

3.1. Overview

Fig. 2 and Algorithm 1 show an overview and the pseudo code of our method, respectively. Our goal is to estimate the error $||\hat{L}(x,x_v) - L(x,x_v)||$ by clustering and sampling VPLs without evaluating L. Our method represents VPLs with a light tree [WFA*05] and the cluster in the light tree is referred to as a *VPL cluster*. From the root node of the light tree, our method estimates $\hat{L}$ by sampling a small number of VPLs from each VPL cluster $\mathbb{C}_i$ in cut $\mathbb{C}$, and determines whether the following criterion is satisfied.

$$||\hat{L} - L|| < \varepsilon \hat{L}, \tag{7}$$

where ε is the relative error threshold specified by the user, where the arguments are omitted for the simplicity. The criterion, how-

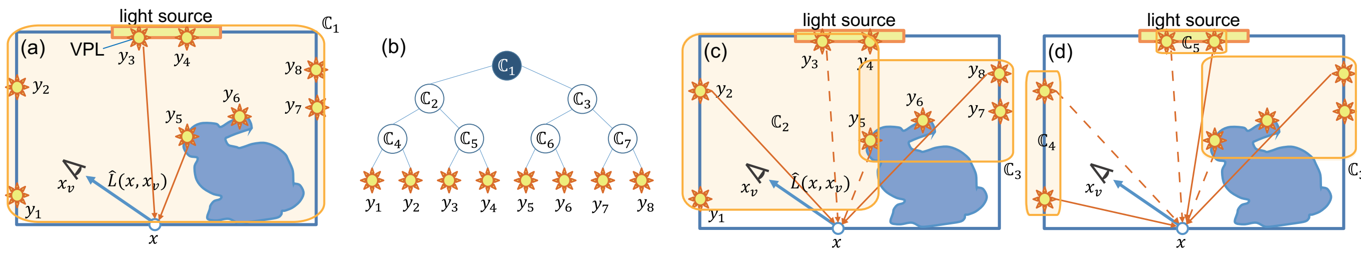

Figure 2: *Overview of our method. (a) Root node of VPL cluster that encompasses all VPLs. (b) Light tree. Each leaf node corresponds to VPL y_i and each inner node corresponds to VPL cluster $\mathbb{C}_i$. From the root node $\mathbb{C}_1$, two VPLs are sampled and $\hat{L}$ and ΔL are calculated. (c) If $\hat{L}$ does not satisfy $\Delta L < \varepsilon \hat{L}$, $\mathbb{C}_1$ is replaced with two child nodes $\mathbb{C}_2$ and $\mathbb{C}_3$. Since y_3 and y_5 are evaluated in the parent node $\mathbb{C}_1$, they are reused and only y_2 and y_8 are sampled for $\mathbb{C}_2$ and $\mathbb{C}_3$, respectively. Then $\hat{L}$ is estimated by summing $\hat{L}_{\mathbb{C}_2}$ and $\hat{L}_{\mathbb{C}_3}$. (d) If $\hat{L}$ does not satisfy the criterion, standard variations σ_2 and σ_3 are calculated and VPL cluster with the largest standard variation, $\mathbb{C}_2$ in this case, is replaced with $\mathbb{C}_4$ and $\mathbb{C}_5$. These processes are repeated until $\hat{L}$ satisfies the criteria.*

ever, contains the unknown (or computationally expensive to evaluate) value of L. To address this problem, our method employs a confidence interval $[\hat{L}-\Delta L, \hat{L}+\Delta L]$ that contains the true value L with probability α which is specified by the user. That is, the probability P_r that the error is less than a tolerance ΔL is α as specified by the following equation.

$$P_r(\|\hat{L}-L\| \leq \Delta L) = \alpha, \tag{8}$$

where the tolerance ΔL is calculated by using the t-distribution [Tho12]. By using the confidence interval, the criterion is calculated by $\Delta L < \varepsilon \hat{L}$ with probability α. To use the confidence interval, our method imposes an additional criterion described in Sec. 3.2. Until the criteria are satisfied, the VPL cluster with the maximum standard variation is selected from the cut nodes and is replaced with its two child nodes.

3.2. Clustering VPLs

To achieve both scalable rendering and efficient sampling for a large number of VPLs, our method employs a stratified sampling, which partitions VPLs into clusters (or strata) of similar VPLs and samples representative VPLs from each cluster. In stratified sampling, if the samples in each cluster follow a normal distribution and the number of samples for each cluster is allocated by using the Neyman allocation [Tho12], statistics T calculated by the following equation follows a t-distribution with $(n-N)$ degrees of freedom.

$$T = \sqrt{\frac{n-N}{n}} \frac{\hat{L}-L}{\sqrt{\sum_{i=1}^{N} s_i^2/n_i}}, \tag{9}$$

where N is the number of clusters, n_i is the number of samples for cluster $\mathbb{C}_i$, n is the total number of samples (i.e. $n = \sum_{i=1}^{N} n_i$), and s_i^2 is the sample variance of $\mathbb{C}_i$. Based on Eq. (9), the tolerance ΔL is calculated using the following equation.

$$\Delta L = t_\alpha \sqrt{\frac{n}{n-N} \sum_{i=1}^{N} \frac{s_i^2}{n_i}}, \tag{10}$$

where t_α is the α quantile of the t-distribution $t(x)$ with $(n-N)$ degrees of freedom that satisfies $\int_{-t_\alpha}^{t_\alpha} t(x)dx = \alpha$.

The rendering efficiency depends on how the VPL clusters are partitioned and how the samples n_i for each VPL cluster are allocated when the total number of samples n is fixed. To this end, we propose an efficient solution for clustering VPLs and allocating samples for each VPL cluster. In stratified sampling, it is known that partitioning into many strata and estimating with a small number of samples is more efficient than a few strata with many samples [PH10]. The variance of estimator in stratified sampling is the sum of variances of each stratum. Therefore, selecting the stratum with maximum standard variation and subdividing the stratum is efficient to reduce the variance. As shown in Eq. (10), the sample variance s_i^2 of each VPL cluster is required to compute the confidence intervals. Since the minimum number of samples to compute the sample variance is two, n_i should be larger than two. As mentioned before, the tolerance ΔL is calculated by using the Neyman allocation, which is the optimal method for allocating samples to each cluster under a fixed number of samples n, as shown in the following equation.

$$n_i = \frac{\sigma_i}{\sum_{k=1}^{N} \sigma_k} n, \tag{11}$$

where σ_i is the standard variation of VPL cluster $\mathbb{C}_i$. By subdividing the VPL cluster with the maximum standard variation repeatedly, the standard variations of VPL clusters approach uniform and the numbers of samples for each VPL cluster also approach uniform as shown in Eq. (11). Then two samples ($n_i = 2$) are allocated for each VPL cluster and n is calculated as $2N$. By substituting $n_i = 2$ and $n = 2N$, the tolerance ΔL simplifies to the following equation.

$$\Delta L = t_\alpha \sqrt{\sum_{i=1}^{N} s_i^2}. \tag{12}$$

Standard Variation Estimation: To select the cluster to subdivide, the standard variation σ_i for $\mathbb{C}_i$ is required. Although σ_i can be estimated by using the sampled variance s_i^2, the estimated standard variation seems poor since only two samples are used. Instead, our method estimates σ_i using Eq. (5). We estimate the variance $\mathrm{Var}[fGV]$ by using the estimated upper bound of BRDF within cluster $\mathbb{C}_i$, $\hat{f}_{ub}(x_v, x, \mathbb{C}_i)$, and the upper bound of the geometry term

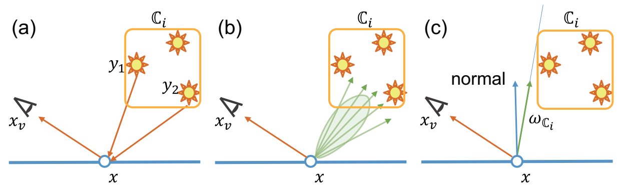

Figure 3: *Estimate the upper bound of BRDF $\hat{f}_{ub}$ by using (a) $f(x_v,x,y_1)$ and $f(x_v,x,y_2)$ which are evaluated at two VPL samples y_1 and y_2, (b) maximum value f^{max} based on BRDF importance sampling, and (c) BRDF $f_{\omega_{\mathbb{C}_i}}$ at direction $\omega_{\mathbb{C}_i}$ that bounds minimum angle between the normal and the VPL cluster.*

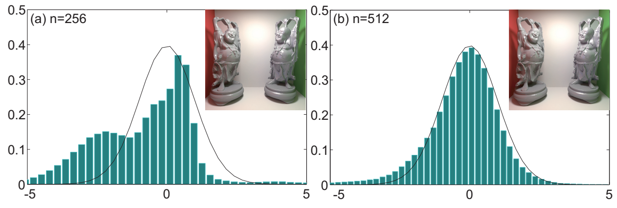

Figure 4: *Histograms of T for $n = 256$ and 512. Horizontal axis represents statistics T and vertical axis represents the probability density. The line shows the t-distribution. Histograms of T are measured in the center pixel of the inset image. By increasing n, T approaches the t-distribution.*

G_{ub} as in the following equation.

$$\sigma_i \approx I_i \hat{f}_{ub}(x_v,x,\mathbb{C}_i)G_{ub}(x,\mathbb{C}_i)\sqrt{\mathrm{Var}[V]}. \tag{13}$$

The variance $\mathrm{Var}[V]$ for $\mathbb{C}_i$ is represented by the average of the visibilities $\bar{V}_i$ within $\mathbb{C}_i$ as given by the following equation.

$$\bar{V}_i = \sum_{y\in\mathbb{C}_i} V(x,y)p(y),$$
$$\mathrm{Var}[V] = \bar{V}_i - \bar{V}_i^2 = -(\bar{V}_i - 1/2)^2 + 1/4, \tag{14}$$

where p is the probability to sample VPL as $p(y) = I(y)/I_i$. Our method conservatively bounds $\sqrt{\mathrm{Var}[V]}$ with the maximum value 0.5, while it can tightly estimate $\mathrm{Var}[V]$ by using the visibility caching described in Sec. 4.

Estimation of the Upper Bound of BRDF: To calculate the standard variation σ_i for each VPL cluster $\mathbb{C}_i$, the upper bound of BRDF within cluster is required. Some BRDF models such as Lambertian, Blinn-Phong, and isotropic Ward BRDF models have a simple and tight way to compute upper bounds as used in Lightcuts [WFA*05]. This limitation is lifted in our method by using BRDF importance sampling and exploiting VPL samples as shown in Fig. 3. BRDF importance sampling [WA09] is performed to estimate $\hat{f}_{ub}$ for glossy BRDFs. Based on BRDF f at shading point x, N_f rays are sampled. Then intersection tests between these rays and the bounding box of the VPL cluster are performed. The maximum value of BRDF f^{max} is calculated from among the intersected rays. As mentioned before, since our method samples two VPLs y_1, y_2, BRDF values $f(x_v,x,y_1)$ and $f(x_v,x,y_2)$ are evaluated and reused for $\hat{f}_{ub}$. In addition, we calculate the minimum angle between the normal at the shading point x and the direction $\omega_{\mathbb{C}_i}$ that intersects the bounding box of the cluster, which is also required in the calculation of G_{ub}. Our method evaluates f by using $\omega_{\mathbb{C}_i}$ as $f_{\omega_{\mathbb{C}_i}}$. The maximum value of f^{max}, $f(x_v,x,y_1)$, $f(x_v,x,y_2)$, and $f_{\omega_{\mathbb{C}_i}}$ is used for $\hat{f}_{ub}$.

Additional Clustering Criterion: As mentioned before, ΔL is derived from the assumption that samples in each cluster follow a normal distribution and the Neyman allocation with uniform standard variations is used. However, it is quite difficult to check whether this assumption is valid or not from samples. Instead, t-distribution is robust to violations of this assumption and Eq. (10) is valid when the number of samples n is sufficiently large and the standard variations σ_i are uniform. We consider that the number of samples is sufficient and the standard variations σ_i are uniform when the criterion $E_{\mathbb{C}_i} < \varepsilon\hat{L}$ used in Lightcuts is satisfied. Since σ_i has the same factors f_{ub} and G_{ub} as in $E_{\mathbb{C}_i}$ as shown in Eqs. (6) and (13), our method utilizes σ_i instead of $E_{\mathbb{C}_i}$ with the criterion

$$\sqrt{2}\sigma_i < \varepsilon\hat{L} \ (\forall\mathbb{C}_i \in \mathbb{C}), \tag{15}$$

where $\sqrt{2}$ is the scaling factor to equate σ_i and $E_{\mathbb{C}_i}$, which stems from the difference of maximum values of $\sqrt{\mathrm{Var}[V]}$ and V_{ub}, and the different number of samples. Fig. 4 shows the histograms of T in Eq. (9) with different n. As shown in Fig. 4, T approaches the t-distribution by increasing n.

In summary, our method partitions VPL clusters until $\Delta L < \varepsilon\hat{L}$ and Eq. (15) are satisfied. If these two criteria are not satisfied, the VPL cluster $\mathbb{C}_k$ with the maximum standard variation σ_k is selected, and $\mathbb{C}_k$ from cut $\mathbb{C}$ is replaced with the left child $\mathbb{C}_L$ and the right child $\mathbb{C}_R$. $\hat{L}_{\mathbb{C}_L}, \hat{L}_{\mathbb{C}_R}, \sigma_L$, and σ_R are estimated by sampling two VPLs and reusing from $\mathbb{C}_k$. Then $\hat{L}$ and ΔL are updated by using $\hat{L}_{\mathbb{C}_L}$ and $\hat{L}_{\mathbb{C}_R}$.

4. Visibility Caching

4.1. Overview

Our method exploits the spatial and directional coherence of visibilities between nearby shading points and VPLs using visibility caching. Shading points are partitioned into clusters referred to as *shading clusters*. Shading clusters are initially partitioned based on the proximities of the normals and positions of the shading points (Fig. 5(a)) in a similar way [WC13]. Several cache points are randomly sampled from the shading points in each shading cluster, and the average of the visibilities between each cache point and each VPL cluster is estimated and recorded as *visibility caches* (Fig. 5(b)). The visibility caches are averaged over all the cache points in each shading cluster (Fig. 5(c)). If the average visibility caches differ from those stored at the cache points, the shading cluster is refined (Fig. 5(d)).

4.2. Calculating the average visibilities and refining the shading clusters

At each cache point c_j, the outgoing radiance $\hat{L}$ is estimated by partitioning the VPL clusters as described in Sec. 3. The average visibility $\hat{V}_i$ between $\mathbb{C}_i$ and c_j is estimated by $\frac{V(c_j,y_i^1)+V(c_j,y_i^2)}{2}$, where

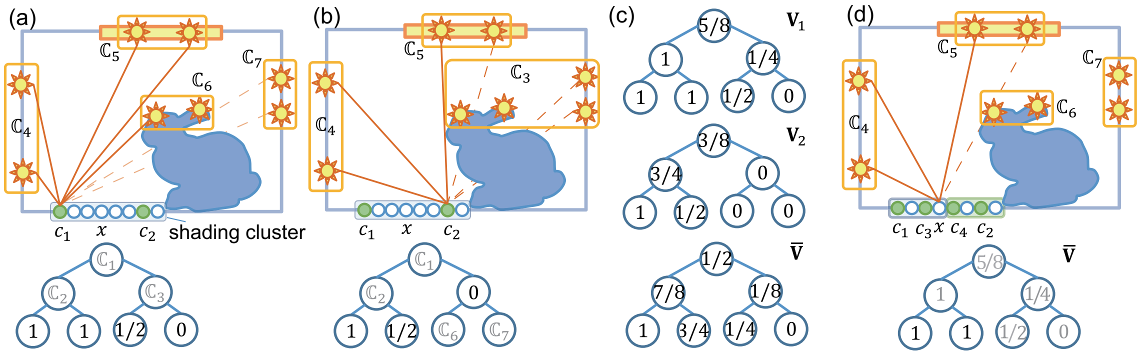

Figure 5: *Visibility caching. (a) The average visibilities between $\mathbb{C}_i$ and cache point c_1 are estimated and recorded at each VPL cluster $\mathbb{C}_i$. (b) The average visibilities between $\mathbb{C}_i$ and c_2. (c) The averaged visibilities at ancestor VPL clusters are estimated, then $\mathbf{V}_1$ and $\mathbf{V}_2$ are constructed. In $\mathbf{V}_2$, average visibilities at descendant nodes $\mathbb{C}_6$ and $\mathbb{C}_7$ are borrowed from that at $\mathbb{C}_3$. $\bar{\mathbf{V}}$ is calculated by averaging $\mathbf{V}_1$ and $\mathbf{V}_2$. δ_1 and δ_2 are calculated. (d) If δ_1 or δ_2 is larger than the threshold δ_{VC}, the shading cluster is subdivided into two clusters.*

y_i^1 and y_i^2 are the sampled VPLs in $\mathbb{C}_i$, respectively. The average visibility $\hat{V}_a$ for the ancestor node $\mathbb{C}_a$ of cut $\mathbb{C}$ is estimated by

$$\hat{V}_a = \left(\sum_{y\in\mathbb{C}_L} p(y)\right)\hat{V}_L + \left(\sum_{y\in\mathbb{C}_R} p(y)\right)\hat{V}_R, \qquad (16)$$

where $\mathbb{C}_L$ and $\mathbb{C}_R$ are the left and right child nodes of $\mathbb{C}_a$, $\hat{V}_L$ and $\hat{V}_R$ are the estimated average visibilities at $\mathbb{C}_L$ and $\mathbb{C}_R$, and p is the probability of sampling the VPLs. By recursively estimating the average visibilities based on the bottom-up approach, the average visibilities at ancestor nodes of VPL clusters are estimated. The visibility cache stores the set of estimated average visibilities for cache point c_j as $\mathbf{V}_j = \{\cdots,\hat{V}_i,\cdots\}$. Please note that our method estimates and records the average visibilities for the ancestor nodes of the VPL clusters only, while previous caching methods [GKPS12, YNI*15] evaluate and store contributions from all the VPLs at each cache point, resulting in expensive computational time and requiring an extensive memory footprint.

Then our method calculates the average $\bar{\mathbf{V}}$ of all the visibility caches $\mathbf{V}$ in the shading cluster. We measure the similarity between $\mathbf{V}_j$ and $\bar{\mathbf{V}}$ by using the following error δ_j.

$$\delta_j = \frac{1}{N_j}\sum_{i=1}^{N_j}\left(\tilde{V}_i - \left(\frac{V(c_j,y_i^1)+V(c_j,y_i^2)}{2}\right)\right)^2, \qquad (17)$$

where N_j is the number of VPL clusters in the cut $\mathbb{C}$ for c_j, $\tilde{V}_i$ is the average value stored in $\bar{\mathbf{V}}$. If δ_j is greater than a threshold δ_{VC}, the caching data $\bar{V}$ is not considered to be adequate for the shading cluster and thus the shading cluster is subdivided into two clusters.

All the shading points in the shading cluster use $\tilde{V}_i$ for $\bar{V}_i$ to estimate $\sqrt{\mathrm{Var}[V]}$ in Eq. (13). If $\tilde{V}_i = 0$ or $\tilde{V}_i = 1$, Var$[V]$ and σ_i become zero, although VPL cluster $\mathbb{C}_i$ has variance in f and G. This leads to $\mathbb{C}_i$ being unpartitioned and introduces errors. To address this problem, our method clamps $\tilde{V}_i$ between 0.01 and 0.9 in order that σ_i cannot to be zero.

5. Results

Figs. 1, 6 to 8 show the rendering results and visualize the relative errors estimated by using our method. The computation times were measured on a PC with an Intel Xeon E5-2697 v2 2.70 GHz CPU. All computations were performed in parallel using multithreading. The image resolution of Figs. 1, 7, and 8 is 1280×720, and that of Fig. 6 is 1024^2. In our experiments, the relative error threshold $\varepsilon = 2\%$ based on Weber's law [WFA*05], $\alpha = 95\%$, number of rays $N_f = 256$, $\delta_{VC} = 0.1$ are used otherwise stated. In visibility caching, ten cache points are used in each shading cluster. We have experimented with various δ_{VC} and the numbers of cache points, but these parameters do not affect the rendering performance and the estimation accuracy so much. Our method measures the rate R_ε of pixels, which satisfy $\|\hat{L}-L\| < \varepsilon L$, in the image. Each VPL cluster stores multiple representative VPLs in a similar way [WABG06]. Our method generates VPLs in the similar way to the traditional many-light rendering method [DKH*13], and the computational times to generate VPLs and light trees are omitted in the following reported computational times since those are common to Lightcuts and our method.

Fig. 1 shows the comparisons of relative error images between (b) our method (282s) and (c) Lightcuts (60s) when $\varepsilon = 2\%$ is specified. Our method achieves $R_\varepsilon = 92.96\%$ while Lightcuts achieves 43.67% in (c). In order to exceed $R_{2\%} = 92.96\%$, ε for the condition $E_{\mathbb{C}_i} < \varepsilon\hat{L}$ of Lightcuts is set to 0.3% with trial and error processes. The rendering time for $\varepsilon = 0.3\%$ is 206s in Lightcuts that estimates $\hat{f}_{ub}$ by using analytical solution, while our method estimates $\hat{f}_{ub}$ by using BRDF importance sampling. To make the comparison fair, our method employs the analytical solution to calculate $\hat{f}_{ub}$, and the computational time T_c is 209s. That is, our method can compete with Lightcuts in rendering performance, while our method can directly render images with reliable accuracy without tedious trial and error processes. We have tried to estimate the sum of the upper bounds of the clustering errors $\sum_{i=1}^{N} E_{\mathbb{C}_i} < 0.02\hat{L}$ for Lightcuts, but it took more than 16 hours (58,078s) and the relative mean error is 5.69214e-06. This indicates that Lightcuts with the estimation of total clustering errors overestimates the error.

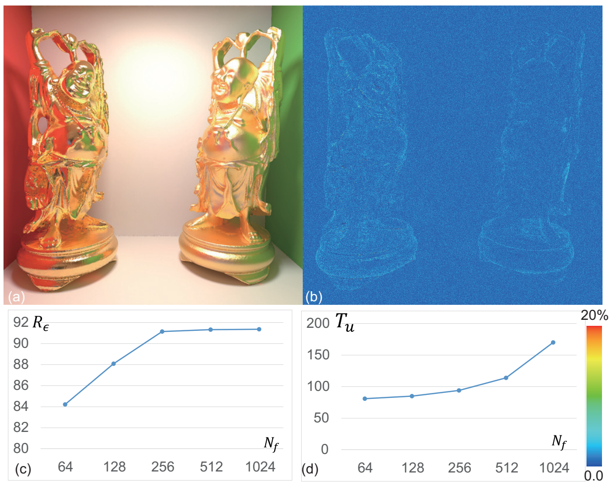

Figure 6: *Buddha scene rendered with Cook-Torrance BRDF illuminated by an area light source and an environment map. (a) rendering result, (b) relative error image, graphs of (c) R_ε and (d) T_u with different N_f. Error bar is shown in the bottom. According to (c) and (d), $N_f = 256$ is used in our experiments.*

Fig. 6 shows the Buddha scene rendered with Cook-Torrance BRDF. By using our method, $R_\varepsilon = 91.15\%$ pixels satisfy the condition $||\hat{L} - L|| < \varepsilon L$ when $\alpha = 95\%$ is specified. Please note that our method bounds the error $||\hat{L} - L||$ stochastically, not deterministically. Therefore, some pixels could have higher relative errors than the specified threshold ε. Figs. 6(c) and (d) show the graphs of R_ε and computational times without visibility caching T_u with different N_f, respectively. As shown in these graphs, R_ε grows by increasing N_f, but it grows slowly when N_f exceeds 256, while the computational times increase considerably. Therefore, our method uses $N_f = 256$ in all experiments.

Table 1: *Scene information of our method. $N_f = 256$, $\varepsilon = 2\%$, $\alpha = 95\%$ are used. N_{vpl} is the number of VPLs, and N_{tri} is that of triangles. T_c and T_u are rendering times in seconds with and without visibility caching, and T_u/T_c indicates the acceleration ratio using the visibility caching. R_ε represents the rate of pixels satisfying $||\hat{L} - L|| < \varepsilon L$ in the image. MRE represents mean relative error.*

Scene	Sponza	Buddha	Kitchen	San Miguel
Fig	Fig. 1	Fig. 6	Fig. 7	Fig. 8
N_{vpl}	2,061k	2,075k	2,082k	1,936k
N_{tri}	262k	1,086k	528k	10,464k
T_c	282	80	70	2,230
R_ε	92.96%	91.15%	91.86%	92.66%
MRE	0.0088	0.0094	0.0091	0.0089
T_u	401	94	86	2,860
R_ε	92.90%	91.19%	92.50%	92.84%
MRE	0.0088	0.0094	0.0089	0.0088
T_u/T_c	1.42	1.18	1.23	1.29

Table 2: *Computational times T_c, T_u, R_ε, and MRE for $\alpha = 99\%$.*

Scene	Sponza	Buddha	Kitchen	San Miguel
T_c	388	103	89	3,151
R_ε	97.67%	95.93%	96.67%	97.94%
MRE	0.0068	0.0075	0.0072	0.0068
T_u	549	126	114	3,872
R_ε	97.68%	95.92%	96.69%	98.06%
MRE	0.0068	0.0075	0.0072	0.0068
T_u/T_c	1.41	1.22	1.28	1.23

Figs. 7 and 8 show the relative error images with different relative error thresholds for scenes with complex geometries and BRDFs. As shown in these images, our method can render images whose errors from the exact solution are stochastically bounded for various relative error thresholds. Our method achieves that R_ε exceeds 91% for all experiments when $\alpha = 95\%$ is specified.

Table 1 shows the number of VPLs N_{vpl}, the computational times T_c with visibility caching and T_u without visibility caching, acceleration ratio T_u/T_c, R_ε, and the mean relative error (MRE) of each image. As shown in Table 1, the visibility caching can speed up the rendering in 18% to 42%, but the decreases in R_ε and the mean relative error are almost negligible. Table 2 shows the computational times T_u, T_c, the rate R_ε, and the mean relative error (MRE) of each scene when $\alpha = 99\%$ is specified. Our method achieves that R_ε exceeds about 96% for all experiments when $\alpha = 99\%$ is specified. The computational times T_c for $\alpha = 99\%$ are up to 1.41 times longer than those for $\alpha = 95\%$ in these examples.

Fig. 9 shows the histogram of relative errors in Buddha scene shown in the inset. Our method achieves $R_\varepsilon = 93.13\%$ when $\alpha = 95\%$ and $\varepsilon = 2\%$ are used. The maximum relative error in Fig. 9 is 20.5%. As shown in Fig. 9, the number of pixels whose relative errors are larger than 2% decreases exponentially. Histograms of the relative errors in other scenes are shown in the supplemental material.

5.1. Discussions

The stochastic evaluation of VPL contributions can introduce noise in the rendered images. Fig. 10 shows the comparison between our method (Figs. 10(a), (b)), Lightcuts (Fig. 10(c)), and the reference (Fig. 10(d)). Relative error threshold ε is set to 2% in our method and Lightcuts. As shown in the zoomed-in images, Fig. 10(c) has perceptible noises since Lightcuts cannot control the relative error of outgoing radiance, while the noises in Fig. 10(b) are imperceptible since our method can directly control the relative error, and due to Weber's law, humans cannot detect the change in less than 2% in practice.

We have applied our visibility caching method to Lightcuts by substituting V_{ub} in Eq. (6) with $2\sqrt{\mathrm{Var}[V]}$. When $\varepsilon = 2\%$ is specified, the computational time and R_ε are 44s and 37.84%, while those without visibility caching are 60s and 43.67%, respectively. Our visibility caching method can accelerate Lightcuts at the cost of decreasing R_ε, while, as we mentioned before, the decrease in R_ε

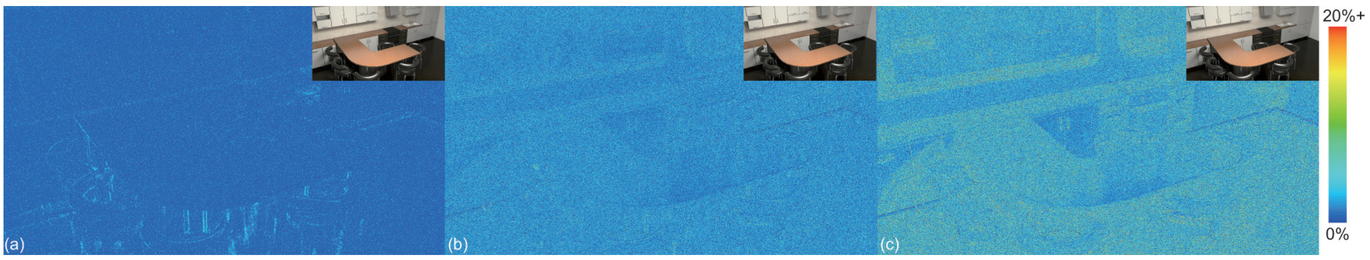

Figure 7: *Kitchen scene rendered with Cook-Torrance, Ashikhmin-Shirely BRDFs. Different relative error thresholds are used. (a) $\varepsilon = 2\%$, $R_\varepsilon = 91.86\%$, and $T_c = 70s$, (b) $\varepsilon = 5\%$, $R_\varepsilon = 93.65\%$, and $T_c = 40s$, (c) $\varepsilon = 10\%$, $R_\varepsilon = 95.30\%$, and $T_c = 31s$. Our method can estimate errors well for various relative error thresholds.*

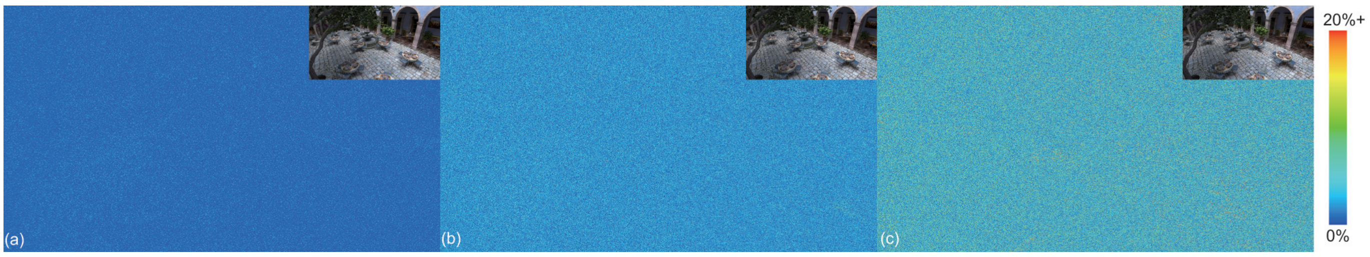

Figure 8: *San Miguel scene rendered with Cook-Torrance, Ashikhmin-Shirely BRDFs. Different relative error thresholds are used. (a) $\varepsilon = 2\%$, $R_\varepsilon = 92.66\%$, and $T_c = 2,230s$, (b) $\varepsilon = 5\%$, $R_\varepsilon = 92.79\%$, and $T_c = 757s$, (c) $\varepsilon = 10\%$, $R_\varepsilon = 92.07\%$, and $T_c = 327s$. Our method can estimate errors well for various relative error thresholds.*

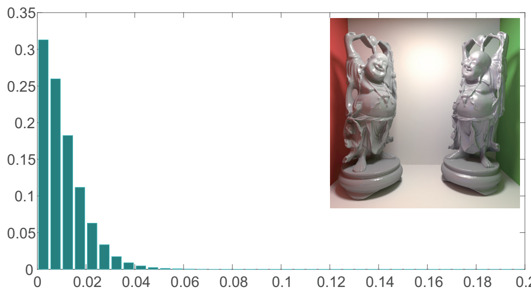

Figure 9: *Histogram of relative errors in Buddha scene. Horizontal axis represents the relative errors, and vertical axis represents the probability of occurrence. $\varepsilon = 2\%$, $R_\varepsilon = 93.13\%$, and $T_c = 86s$.*

of our method is almost negligible. This indicates that our visibility caching method is well suited to our error estimation framework.

Since our method is based on many-light rendering, our method also inherits the drawbacks of many-light rendering, such as the singularities due to the geometry term and highly glossy BRDFs. These singularities lead to splotches in the rendered images, but the splotches are alleviated by increasing the number of VPLs or clamping. Due to the scalability of our method, a large number of VPLs as shown in Table 1 can be handled very efficiently.

6. Conclusions and Future Work

We have proposed an error estimation framework for many-light rendering by using confidence intervals. By specifying the relative error threshold ε and confidence parameter α, our method can automatically partition VPLs so that the error of the outgoing radiance is stochastically bounded. We demonstrated that our method can render images with reliable accuracy for complex scenes and various materials. In addition, we introduced an efficient visibility caching method that can accelerate rendering 18% to 42% without a noticeable decrease in the estimation accuracy.

In future work, we would like to extend our method to Multidimensional Lightcuts [WABG06]. Our method can be combined with Multidimensional Lightcuts by estimating the errors using clusters of the light tree and the gather tree, but an efficient method to compute the upper bounds of phase functions and BRDFs for gather points and VPLs is required. Furthermore, we would like to estimate errors for more complex lighting effects (e.g. caustics).

Acknowledgements

This research was partially supported by JSPS KAKENHI, Grant-in-Aid for Scientific Research on Innovative Areas, Grant Number JP15H05924.

References

[AUT13] AUTODESK: Autodesk 360 rendering, 2013. URL: `http://rendering.360.autodesk.com/`. 2

[BMB15] BUS N., MUSTAFA N. H., BIRI V.: Illuminationcut. *Computer Graphics Forum 34*, 2 (2015), 561–573. 2

[CAM08] CLARBERG P., AKENINE-MOLLER T.: Exploiting visibility correlation in direct illumination. *Computer Graphics Forum 27*, 4 (2008), 1125–1136. 2

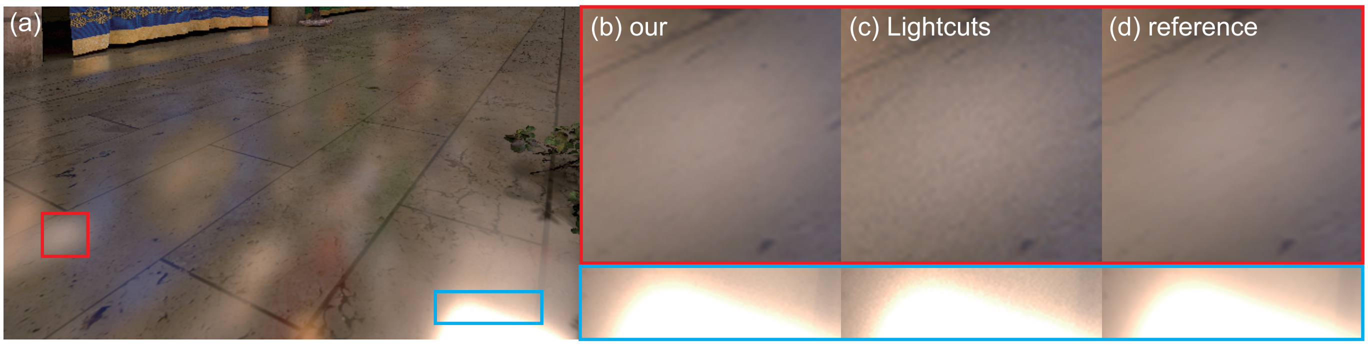

Figure 10: *Comparison between our method, Lightcuts, and the reference. Zoomed-in images show that the noises due to (b) our stochastic evaluation are imperceptible and indistinguishable from (d) the reference, while (c) those in Lightcuts are perceptible.*

[DHKL10] DAMMERTZ H., HANIKA J., KELLER A., LENSCH H.: A hierarchical automatic stopping condition for Monte Carlo global illumination. In *Proceedings of the Winter School of Computer Graphics* (2010), pp. 159–164. 3

[DKH*10] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P., BALA K.: Combining global and local virtual lights for detailed glossy illumination. *ACM Transactions on Graphics 26*, 6 (2010), 143:1–143:8. 2

[DKH*13] DACHSBACHER C., KRIVANEK J., HASAN M., ARBREE A., WALTER B., NOVAK J.: Scalable realistic rendering with many-light methods. In *Eurographics 2013 - State of the Art Reports* (2013). 2, 6

[GKPS12] GEORGIEV I., KŘIVÁNEK JAROSLAV J., POPOV S., SLUSALLEK P.: Importance caching for complex illumination. *Computer Graphics Forum 31*, 2 (2012), 701–710. 2, 6

[HJJ10] HACHISUKA T., JAROSZ W., JENSEN H. W.: A progressive error estimation framework for photon density estimation. *ACM Transactions on Graphics 29*, 6 (2010), 144:1–144:12. 3

[HKWB09] HAŠAN M., KŘIVÁNEK J., WALTER B., BALA K.: Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics 28*, 5 (2009), 143:1–143:6. 2

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics 26*, 3 (2007), 26:1–26:10. 2

[HWJ*15] HUO Y., WANG R., JIN S., LIU X., BAO H.: A matrix sampling-and-recovery approach for many-lights rendering. *ACM Transactions on Graphics 34*, 6 (2015), 210:1–210:12. 2

[Kel97] KELLER A.: Instant radiosity. In *Proc. of SIGGRAPH '97* (1997), pp. 49–56. 2

[KGKC13] KŘIVÁNEK J., GEORGIEV I., KAPLANYAN A., CANAD J.: Recent advances in light transport simulation: Theory and practice. In *ACM SIGGRAPH 2013 Courses* (2013). 2

[MJL*13] MOON B., JUN J. Y., LEE J., KIM K., HACHISUKA T., YOON S.-E.: Robust Image Denoising Using a Virtual Flash Image for Monte Carlo Ray Tracing. *Computer Graphics Forum* (2013). 3

[OP11] OU J., PELLACINI F.: Lightslice: Matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics 30*, 6 (2011), 179:1–179:8. 2

[PGSD13] POPOV S., GEORGIEV I., SLUSALLEK P., DACHSBACHER C.: Adaptive quantization visibility caching. *Computer Graphics Forum 32*, 2 (2013). 2

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., 2010. 4

[Pur87] PURGATHOFER W.: A statistical method for adaptive stochastic sampling. *Computer & Graphics 11*, 2 (1987), 157–162. 3

[RFS03] RIGAU J., FEIXAS M., SBERT M.: Refinement Criteria Based on f-Divergences. In *Eurographics Workshop on Rendering* (2003). 3

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics 27*, 5 (2008), 129:1–129:8. 2

[SHD15] SIMON F., HANIKA J., DACHSBACHER C.: Rich-vpls for improving the versatility of many-light methods. *Computer Graphics Forum (Proceedings of Eurographics) 34*, 2 (2015), 575–584. 2

[Tho12] THOMPSON S. K.: *Sampling*, 3rd ed. Wiley, 2012. 4

[TJ97] TAMSTORF R., JENSEN H. W.: Adaptive sampling and bias estimation in path tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97* (1997), pp. 285–296. 3

[Tok15] TOKUYOSHI Y.: Virtual spherical gaussian lights for real-time glossy indirect illumination. *Computer Graphics Forum 34*, 7 (2015), 89–98. 2

[UNRD13] ULBRICH J., NOVAK J., REHFELD H., DACHSBACHER C.: Progressive visibility caching for fast indirect illumination. In *Proc. of International Workshop on Vision, Modeling, and Visualization* (2013), p. 8. 2

[WA09] WANG R., AKERLUND O.: Bidirectional importance sampling for unstructured direct illumination. *Computer Graphics Forum 28*, 2 (2009), 269–278. 2, 5

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics 25*, 3 (July 2006), 1081–1088. 2, 6, 8

[WC13] WU Y.-T., CHUANG Y.-Y.: Visibilitycluster: Average directional visibility for many-light rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1566–1578. 2, 5

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics 24*, 3 (2005), 1098–1107. 1, 2, 3, 5, 6

[WHY*13] WANG R., HUO Y., YUAN Y., ZHOU K., HUA W., BAO H.: Gpu-based out-of-core many-lights rendering. *ACM Transactions on Graphics 32*, 6 (2013), 210:1–210:10. 2

[WKB12] WALTER B., KHUNGURN P., BALA K.: Bidirectional lightcuts. *ACM Transactions on Graphics 31*, 4 (2012), 59:1–59:11. 2

[YNI*15] YOSHIDA H., NABATA K., IWASAKI K., DOBASHI Y., NISHITA T.: Adaptive importance caching for many-light rendering. *Journal of WSCG 23*, 1 (2015), 65–72. 2, 6