A Progressive Error Estimation Framework for Photon Density Estimation

Toshiya Hachisuka* Wojciech Jarosz†,* Henrik Wann Jensen*

*UC San Diego †Disney Research Zürich

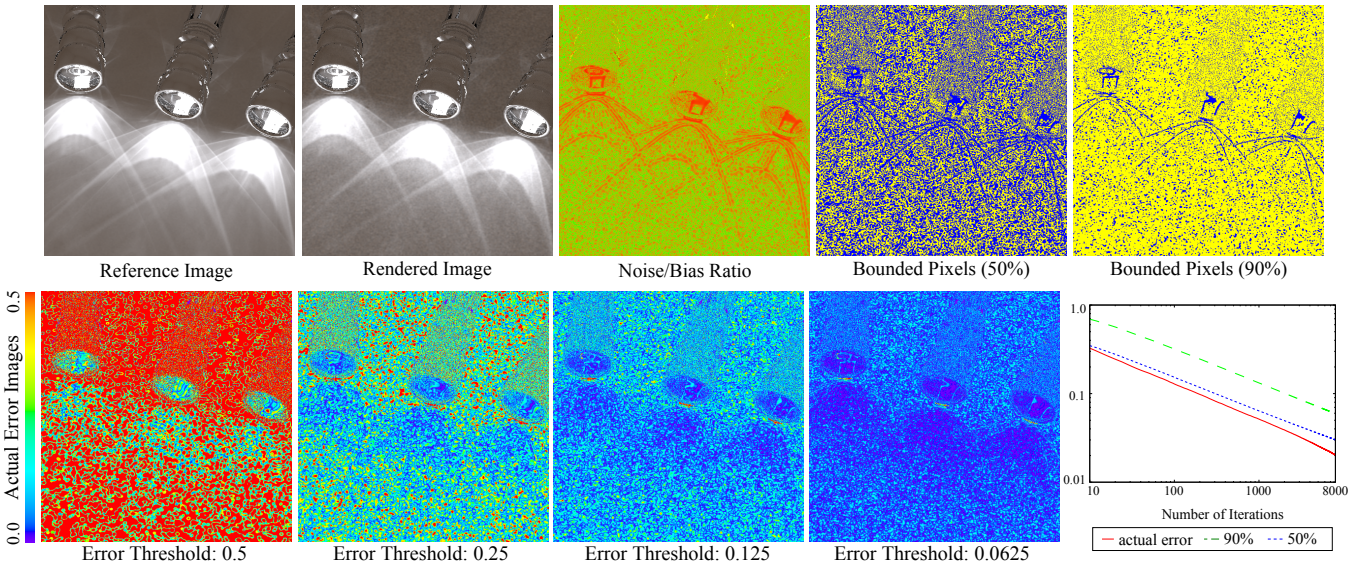

Figure 1: *Our error estimation framework takes into account error due to noise and bias in progressive photon mapping. The reference image is rendered using more than one billion photons, and the rendered image is the result using 15M photons. The noise/bias ratio shows the areas of the image dominated by noise (in green) or bias (in red). The bounded pixels images show the results corresponding to a desired confidence in the estimated error value (pixels with bounded error are shown yellow): a higher confidence (90%) provides a conservative per pixel error estimate, and lower confidence (50%) is useful to estimate the average error. One application of our error estimation framework is automatic rendering termination with a user-specified error threshold (bottom row: the images show color-coded actual error). Our framework estimates error without any input from a reference image.*

Abstract

We present an error estimation framework for progressive photon mapping. Although estimating rendering error has been established for unbiased rendering algorithms, error estimation for biased rendering algorithms has not been investigated well in comparison. We characterize the error by the sum of a bias estimate and a stochastic noise bound, which is motivated by stochastic error bounds formulation in biased methods. As a part of our error computation, we extend progressive photon mapping to operate with smooth kernels. This enables the calculation of illumination gradients with arbitrary accuracy, which we use to progressively compute the local bias in the radiance estimate. We also show how variance can be computed in progressive photon mapping, which is used to estimate the error due to noise. As an example application, we show how our error estimation can be used to compute images with a given error threshold. For this example application, our framework only requires the error threshold and a confidence level to automatically terminate rendering. Our results demonstrate how our error estimation framework works well in realistic synthetic scenes.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

Keywords: Photon Mapping, Density Estimation

1 Introduction

Global illumination algorithms are increasingly used for predictive rendering to verify lighting and appearance of a given scene. In industrial design, rendered images are used to predict the appearance of new products, architects rely on rendered images to guide the design of new buildings, lighting engineers use global illumination to verify the lighting levels in buildings, and lamp manufacturers use global illumination to predict the lighting patterns from new lamps. Since global illumination is used to compute lighting and make decisions based on the computed values, it is critical that the accuracy of the final result is known to the user. In other words, we need error estimation of the final result.

Unbiased Monte Carlo ray tracing algorithms are commonly assumed to be the only choice when error estimation is required. However, unbiased methods are not necessarily the optimal choice

for accurate lighting simulation. For example, biased methods are currently considered to be the only technique to robustly handle specular reflection or refraction of caustics from small light sources [Hachisuka et al. 2008]. These paths are common in light transport within realistic optical systems such as light bulbs and headlights, thus it is highly desirable to have error estimation for biased methods. Moreover, since biased methods, such as photon mapping [Jensen 2001], are commonly used in commercial and industrial rendering systems, investigating error estimation for biased methods will also be useful for a wide range of applications.

Our contribution in this paper is a new error estimation framework for progressive photon mapping. We derive an error estimation framework by constructing a bias estimator and noise estimator in progressive photon mapping. We demonstrate that our estimated error captures error due to noise and bias under complex light transport. Although error estimation itself is already of high value, as an example application, we also demonstrate that it can be used to automatically terminate rendering without any subjective trial and error by the user. Our error estimation induces very little computational overhead, resulting in negligible increases in render time. The user can specify a desirable confidence to either obtain per pixel error estimates or an average error estimate across the image. Figure 1 highlights our results.

2 Related Work

It is well known that error estimation in unbiased Monte Carlo ray tracing algorithms, such as path tracing [Kajiya 1986] and bidirectional path tracing [Lafortune and Willems 1993; Veach and Guibas 1995], can be done by computing the variance of the solution. This fact that error in unbiased Monte Carlo methods appears as variance has been used in many applications. For example, variance is often used for adaptive sampling in order to find regions with large error [Lee et al. 1985; Purgathofer 1987; Tamstorf and Jensen 1997]. Assuming errors are distributed according the normal distribution, it is also possible to approximate a stochastic error bound based on variance, which provides an estimated range of error based on a confidence probability provided by the user. Unfortunately, the same technique cannot be used as an error estimate for biased Monte Carlo methods, such as photon mapping [Jensen 2001], since these methods suffer from *bias*.

Although error estimation in biased methods in general has not been well-studied compared to unbiased methods, there has been some effort to derive error estimators for biased methods. An example includes the error bound used to guide the hierarchical sampling of many point lights in Lightcuts [Walter et al. 2005] and the first-order Taylor expansion of error in a hypothetical Lambertian "split-sphere" environment used by irradiance caching [Ward et al. 1988].

If we focus on photon density estimation, which is our focus as well, Myszkowski [1997] proposed to use multiple radii during radiance estimation to estimate bias, and then selected the estimate that minimized an error heuristic. Walter [1998] improved this heuristic error estimation using a perceptual metric. Bias compensation [Schregle 2003] also uses multiple radii around the same point, but uses a binary search to find the radius with the minimum heuristic error. This method estimates bias by assuming the radiance estimate with the smallest radius is an unbiased estimate. These prior techniques all use similar error analyses as ours, however, none of them provide an actual estimate of error as is possible with unbiased methods. Our method provides an algorithm to estimate error, especially focusing on progressive photon mapping.

In order to estimate bias, our method estimates derivatives of the illumination using kernel derivatives in radiance/density estimation. The same idea is commonly used in standard density estimation methods [Silverman 1986]; however, the key difference is that our derivation is based on progressive density estimation proposed by Hachisuka et al [2008]. As with progressive density estimation, our derivative estimates converge to the correct solution with a bounded memory consumption. This is not the case for standard density estimation methods, which require storing an infinite number of samples to obtain the correct solution. Gradient computation of light transport itself has been studied in previous work [Ward and Heckbert 1992; Ramamoorthi et al. 2007; Jarosz et al. 2008], but we introduce an alternative method which integrates easily in our error estimation framework.

There are several techniques that reduce the bias of photon density estimation [Hey and Purgathofer 2002; Havran et al. 2005; Herzog et al. 2007]. Although it is conceivable to use our bias estimate for bias reduction, we do not target this application in our paper; instead, our goal is to estimate the bias with reasonable accuracy for error estimation in rendering.

3 Progressive Photon Mapping

Since our error estimation is based on progressive photon mapping, we briefly outline the algorithm in this section. Progressive photon mapping [Hachisuka et al. 2008] solves the rendering equation using a multi-pass approach. The first pass consists of ray tracing the scene and storing all the non-specular visible hit points seen along each ray path in a data structure of *measurement points*.

The subsequent passes perform photon tracing. During each photon tracing pass, statistics about the photons are accumulated by the measurement points. These accumulated statistics refine radiance estimates at the measurement points in such a way that the radiance estimate is consistent and therefore converges to the correct solution in the limit.

The primary statistics that each measurement point maintains include the current search radius R, the accumulated photon count N, and the reflected flux τ. Note that Hachisuka et al. used τ to express *un*-normalized accumulated flux; however, to obtain a consistent notation within this paper, we instead use τ to express *normalized* flux. We summarize our notation in Table 1.

Symbol	Description
$\mathbf{x}$	Position of a measurement point
$\vec{\omega}$	Outgoing direction of a measurement point
N^e	Number of emitted photons in each pass
N_i^e	Total number of emitted photons after i passes
N_i	Accumulated intercepted photon count at $\mathbf{x}$ after pass i
M_{i+1}	Number of intercepted photons at $\mathbf{x}$ during pass $i+1$
R_i	Radius at photon pass i
α	Fraction of photons kept after each pass
$\tau_i(\mathbf{x}, \vec{\omega})$	Flux within $R_i(\mathbf{x})$ after pass i
$\tilde{\tau}_{i+1}(\mathbf{x}, \vec{\omega})$	Flux within $R_i(\mathbf{x})$ using only photons from pass i
$\tau'_{i+1}(\mathbf{x}, \vec{\omega})$	Flux within $R_i(\mathbf{x})$ in pass $i+1$ (*before* radius reduction)
$\tau_{i+1}(\mathbf{x}, \vec{\omega})$	Flux within $R_{i+1}(\mathbf{x})$ after pass $i+1$ (*after* radius reduction)

Table 1: *Definitions of quantities used throughout this paper.*

Each measurement point tracks the photons which fall within its search radius R and accumulates their power into τ. After pass i, the radiance at a measurement point is approximated as:

$$L_i(\mathbf{x}, \vec{\omega}) \approx \frac{\tau_i(\mathbf{x}, \vec{\omega})}{\pi R_i(\mathbf{x})^2}. \quad (1)$$

After the first photon tracing pass, $\tau_1(\mathbf{x}, \vec{\omega})$ stores the flux premulti-

plied by the BRDF from all N_1 photons landed within radius R_1:

$$\tau_1(\mathbf{x}, \vec{\omega}) = \frac{1}{N_1^e} \sum_{p=1}^{N_1} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p) \Phi_p(\mathbf{x}_p, \vec{\omega}_p), \tag{2}$$

where $\vec{\omega}$ is the outgoing direction at the measurement point, $\vec{\omega}_p$ is the incoming direction of the photon, f_r is the BRDF, and $\Phi(\mathbf{x}_p, \vec{\omega}_p)$ is the flux carried by photon p. The number of photons *emitted* in each pass is N^e, and we use $N_i^e = iN^e$ to express the *total* number of emitted photons up to and including pass i. Note that the radius R_1 and number of *intercepted* photons N_1 both depend on the measurement point $\mathbf{x}$, but we omit these dependencies in our notation for simplicity.

After the first pass, these statistics are refined by performing additional photon tracing passes, each emitting N^e more photons. During pass $i+1$, M_{i+1} additional photons are intercepted at $\mathbf{x}$. The flux from just these additional photons would be:

$$\tilde{\tau}_{i+1}(\mathbf{x}, \vec{\omega}) = \frac{1}{N^e} \sum_{p=1}^{M_{i+1}} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p) \Phi_p(\mathbf{x}_p, \vec{\omega}_p). \tag{3}$$

If we used all M_{i+1} photons and kept the radius fixed, we could obtain an improved estimate for the flux using all $N_i + M_{i+1}$ photons by just accumulating this additional flux as:

$$\begin{aligned} \tau'_{i+1}(\mathbf{x}, \vec{\omega}) &= \frac{1}{N_{i+1}^e} \sum_{p=1}^{N_i + M_{i+1}} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p) \Phi_p(\mathbf{x}_p, \vec{\omega}_p), \\ &= \frac{N_i^e \tau_i + N^e \tilde{\tau}_{i+1}}{N_{i+1}^e} = \frac{i\tau_i + \tilde{\tau}_{i+1}}{i+1}. \end{aligned} \tag{4}$$

However, to obtain a consistent estimator, progressive photon mapping must ensure that the bias and the variance are reduced at each iteration. This is accomplished by reducing the radius and statistically keeping only a fraction of the intercepted photons at each iteration. A single parameter $\alpha \in (0, 1)$ is used to control the fraction of photons to keep from each pass. If M_{i+1} photons are found within search radius R_i during photon pass $i+1$, the new accumulated photon count is computed as:

$$N_{i+1} = N_i + \alpha M_{i+1}, \tag{5}$$

and the reduced radius is computed as:

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_{i+1}}{N_i + M_{i+1}}}. \tag{6}$$

Due to this radius reduction, we must account for the flux of the photons that now fall outside the reduced radius. To account for this, the accumulated flux at the next iteration is computed as:

$$\tau_{i+1}(\mathbf{x}, \vec{\omega}) = \tau'_{i+1}(\mathbf{x}, \vec{\omega}) \frac{{R_{i+1}}^2}{{R_i}^2} = \tau'_{i+1}(\mathbf{x}, \vec{\omega}) \frac{N_i + \alpha M_{i+1}}{N_i + M_{i+1}}. \tag{7}$$

Traditional photon mapping can be considered as a special case of progressive photon mapping where the radius $R_i(\mathbf{x})$ stays the same ($\alpha = 1.0$).

4 Error Estimation Framework

It is well known that stochastic error bounds in unbiased Monte Carlo rendering methods can be estimated using the variance of the estimate[1]. The reason for this is that the error is caused purely by *noise* due to stochastic sampling. However, biased Monte Carlo rendering methods have *bias* in addition to noise. Bias is the difference between the estimated radiance without noise and the correct radiance. Since photon density estimation, including progressive photon mapping, is a biased rendering method, we need to take into account both bias and noise in the error estimate.

In biased methods, the error can be expressed as the sum of bias B_i of the estimate and noise N_i due to sampling (we denote the exact bias as B_i since bias is changing at each iteration in progressive photon mapping):

$$L_i - L = B_i + N_i, \tag{8}$$

where L_i is the estimated radiance and L is the true radiance. This *bias-noise* decomposition has been used in prior work (such as [Schregle 2003]) as well and it is applicable to any biased Monte Carlo methods, including progressive photon mapping. Now, assuming we know bias B_i and sample variance V_i, the central limit theorem states that for an infinite number of samples:

$$\frac{L_i - L - B_i}{\sqrt{V_i}} \xrightarrow{d} N(0, 1), \tag{9}$$

where we use $\xrightarrow{d}$ to denote that the *distribution* of the left-hand-side converges to the standard normal distribution $N(0, 1)$. Note that this does not directly estimate error, but rather estimates the error distribution. Although this relationship is valid only for an infinite number of samples, it has been shown that assuming this is valid for a finite number of samples is reasonable in practice [Tamstorf and Jensen 1997; Fiorio 2004]. This also indicates that the variance-based error estimation in unbiased methods is fundamentally an approximation unless we use an infinite number of samples. Our error estimation framework is also based on a few approximations as we will describe throughout the paper.

Based on the error distribution, we can derive a stochastic bound (confidence interval) of radiance estimation using the t-distribution:

$$\begin{aligned} &P\left(-E_i < L_i - L - B_i < E_i\right) = 1 - \beta \\ &E_i(\mathbf{x}) = t\left(i, 1 - \frac{\beta}{2}\right) \sqrt{\frac{V_i(\mathbf{x})}{i}}, \end{aligned} \tag{10}$$

where $t\left(i, 1 - \frac{\beta}{2}\right)$ is the $1 - \frac{\beta}{2}$ percentile of the t-distribution with degree i. The t-distribution describes the distribution of the quantity $\frac{L_i - L - B_i}{\sqrt{iV_i}}$. This gives the interval $[-E_i, E_i]$ that contains $L_i - L - B_i$ with probability $1 - \beta$. Note that this is a well-known derivation and interested readers should refer to, for example, Geertsema [1970]. It is also important to note that the interval $[-E_i, E_i]$ is not for "error", but rather for "error - bias" at this point.

In order to estimate the interval for error, we shift both the value $L_i - L - B_i$ and the interval $[-E_i, E_i]$ by adding B_i:

$$\begin{aligned} &P\left(-E_i < L_i - L - B_i < E_i\right) \\ &= P\left(-E_i + B_i < L_i - L < E_i + B_i\right). \end{aligned} \tag{11}$$

[1] Note that there is a critical difference between a *stochastic* bound and a *deterministic* bound. A deterministic bound always correctly bounds the value, but a stochastic bound only correctly bounds with some probability. This probability is usually provided by the user as a desirable *confidence* in the bound. Standard variance-based methods in fact compute a stochastic bound and, unfortunately, there is little hope of obtaining a deterministic bound for Monte Carlo estimation. More information on this concept can be found in statistics textbook such as Hastie et al. [2001].

Note that the inequalities inside are equivalent. This operation is valid because bias B_i, by definition, is a deterministic variable since bias is a difference between an expected value and the correct value which are both deterministic. In other words, bias just affects error as a uniform, constant shift of the error distribution.

The above signed confidence-interval can be used as-is, however, it is sometimes desirable to obtain a single value as the stochastic bound of absolute error. In order to obtain such an expression, we extend the interval by taking the absolute value of the bias:

$$\begin{aligned} &P\left(-E_i + B_i < L_i - L < E_i + B_i\right) \\ &\leq P\left(-E_i - |B_i| < L_i - L < E_i + |B_i|\right) \\ &= P\left(|L_i - L| < E_i + |B_i|\right), \end{aligned} \tag{12}$$

which provides $E_i + |B_i|$ as a single-valued stochastic error bound of the absolute error. This theoretically gives us a confidence interval with larger confidence than the user-specified confidence $1-\beta$. Note that the above equations do not use the bound of bias, but bias itself.

Dependence of error on the absolute intensity of radiance is often not desirable since the human visual system is more sensitive to contrast than absolute differences in intensity [Glassner 1995]. Therefore, we divide the interval by the estimated radiance to obtain the relative stochastic error bound:

$$P\left(|L_i - L| < E_i + |B_i|\right) = P\left(\left|\frac{L_i - L}{L_i}\right| < \frac{E_i + |B_i|}{L_i}\right). \tag{13}$$

The interval is $\left[0, \frac{E_i+|B_i|}{L_i}\right]$ with confidence larger than $1-\beta$. Note that the above Equation 8 to Equation 13 are theoretically valid only if we know the exact bias and the exact sampled variance.

The challenge here is that both the exact bias B_i and the exact sample variance V_i are unknown. We therefore need to rely on estimations of bias and variance. Estimation of bias B_i has been investigated by previous work, but none of them can be extended to progressive photon mapping. Moreover, we do not have an estimator of variance V_i in progressive photon mapping. To be concrete, progressive density estimation has a different formulation than standard radiance estimation and samples are correlated due to radius reduction. We therefore have to consider the correlation between samples when estimating variance. The key contributions of our work are estimators of B_i and V_i which we describe in the following sections. Note that, in practice, we cannot obtain the exact stochastic bounds as defined above due to approximations in bias and variance estimators.

4.1 Bias Estimation

We first show that the bias in progressive radiance estimation is actually the average of bias induced by the kernel radii at each iteration

$$B_i = \frac{1}{i}\sum_{j=1}^{i} B_{p,j}, \tag{14}$$

where $B_{p,j}$ is bias of the radiance estimate using only the photons from the j^{th} pass. We then introduce a bias estimator used in standard density estimation, which is applied for progressive photon mapping.

To obtain Equation 14, we start by expanding the estimated radiance $L_{j+1}(\mathbf{x})$ at the $j+1^{\text{th}}$ iteration using the refinement procedure described in Section 3:

$$\begin{aligned} L_{j+1}(\mathbf{x}) &= \frac{\tau_{j+1}(\mathbf{x},\vec{\omega})}{\pi R_{j+1}(\mathbf{x})^2} = \frac{R_{j+1}(\mathbf{x})^2}{R_j(\mathbf{x})^2}\frac{\tau'_{j+1}(\mathbf{x},\vec{\omega})}{\pi R_{j+1}(\mathbf{x})^2} \\ &= \frac{j\tau_j(\mathbf{x},\vec{\omega}) + \tilde{\tau}_{j+1}(\mathbf{x},\vec{\omega})}{j+1}\frac{1}{\pi R_j(\mathbf{x})^2} \\ &= \frac{1}{j+1}\left(j\frac{\tau_j(\mathbf{x},\vec{\omega})}{\pi R_j(\mathbf{x})^2} + \frac{\tilde{\tau}_{j+1}(\mathbf{x},\vec{\omega})}{\pi R_j(\mathbf{x})^2}\right) \\ &= \frac{1}{j+1}\left(jL_j(\mathbf{x}) + L_{p,j}(\mathbf{x})\right), \end{aligned} \tag{15}$$

where $L_{p,j}(\mathbf{x})$ is radiance estimate using only the photons from the j^{th} pass. Note that $\frac{\tilde{\tau}_{j+1}(\mathbf{x},\vec{\omega})}{\pi R_j(\mathbf{x})^2} = L_{p,j}(\mathbf{x})$ because $\tilde{\tau}_{j+1}(\mathbf{x},\vec{\omega})$ includes the intercepted photons only at the $j+1^{\text{th}}$ iteration *before* the radius reduction and the radius reduction happens after flux accumulation (i.e., the radius used in the photon query is $R_j(\mathbf{x})$).

The key observation is that $L_{p,j}$ is equivalent to the radiance estimate using a standard radiance estimation with radius R_j because it uses photons only from the j^{th} pass. Therefore, we can write down $E[L_{p,j}] = L(\mathbf{x}) + B_{p,j}$ based on the definition of bias in standard radiance estimation. Using Equation 15 recursively, the progressive radiance estimate $L_i(\mathbf{x})$ is expressed as:

$$\begin{aligned} E\left[L_i(\mathbf{x})\right] &= E\left[\frac{1}{i}\sum_{j=1}^{i} L_{p,j}(\mathbf{x})\right] = L(\mathbf{x}) + B_i \\ &= \frac{1}{i}\sum_{j=1}^{i}\left(L(\mathbf{x}) + B_{p,j}\right) = L(\mathbf{x}) + \frac{1}{i}\sum_{j=1}^{i} B_{p,j}. \end{aligned} \tag{16}$$

Finally, subtracting $L(\mathbf{x})$ from both sides gives us Equation 14.

Silverman [1986] showed that for standard density estimation bias can be approximated using the Laplacian of the density. We can apply this to each bias value $B_{p,j}$ as

$$\begin{aligned} B_{p,j} &= \frac{1}{2}R_j(\mathbf{x})^2 k_2 \nabla^2 L_{p,j}(\mathbf{x}) + O\left(\sum_k \frac{\partial^k L_{p,j}(\mathbf{x}_k)}{\partial \mathbf{x}_k^k} R_j(\mathbf{x})^4\right) \\ &\approx \frac{1}{2}R_j(\mathbf{x})^2 k_2 \nabla^2 L_{p,j}(\mathbf{x}), \end{aligned} \tag{17}$$

where $k_2 = \int t^2 K(t)\vec{dt}$ is a constant derived from the kernel $K(t)$ (see Appendix A for details). Note that we drop the residual term proportional to $R_j(\mathbf{x})^4$ in our estimation, thus our bias estimator always estimates an approximation of the true bias.

Unfortunately, we cannot directly use this approximation for progressive photon mapping. This approximation needs to estimate $\nabla^2 L_{p,j}(\mathbf{x})$ at each iteration and combine Equation 17 and Equation 14 to compute the Laplacian of the progressive estimate $\nabla^2 L_i(\mathbf{x})$. However, estimating and storing $\nabla^2 L_{p,j}(\mathbf{x})$ at each iteration in progressive radiance estimation is not tractable since this would require unbounded memory and we use a relatively small number of photons per iteration. Therefore, we propose the following approximation of the Laplacian of the progressive estimate $\nabla^2 L_i(\mathbf{x})$ which eliminates the need to store the Laplacian of each iteration.

First, substituting Equation 17 into Equation 14, we obtain

$$B_i = \frac{1}{i}\sum_{j=1}^{i} B_{p,j} \approx \frac{1}{i}\sum_{j=1}^{i}\frac{1}{2}R_j(\mathbf{x})^2 k_2 \nabla^2 L_{p,j}(\mathbf{x}). \tag{18}$$

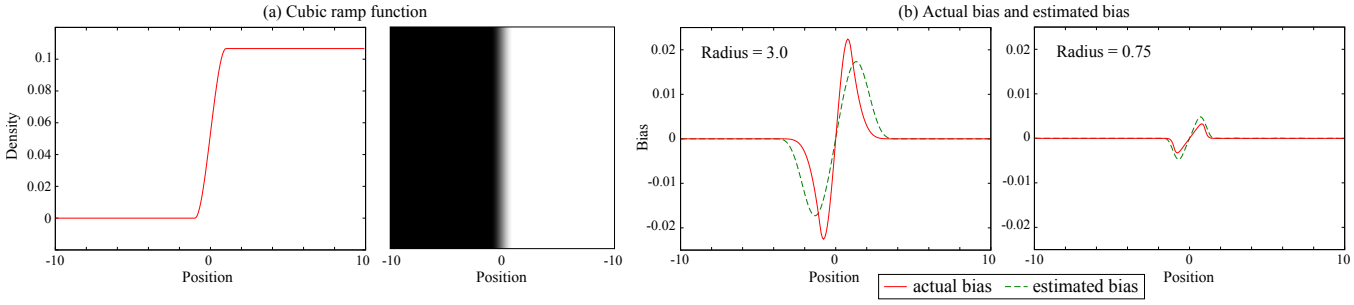

Figure 2: *The graph and image in (a) show the cubic ramp function* $(3t^2 - 2t^3$ *with* $t = clamp(\frac{x-1}{2})$ *where* $y = clamp(x)$ *clamps* x *into* $[0, 1]$*). The graphs in (b) show comparisons of the estimated bias and the actual bias in the cubic ramp scene. The bias estimate uses radius* 3.0 *and* 0.75*. The RMS errors of the bias estimation are* 3.751×10^{-3} *and* 4.746×10^{-4}.

We approximate the summation as (note that we use $R_i(\mathbf{x})$ not $R_j(\mathbf{x})$)

$$\approx \frac{1}{2} R_i(\mathbf{x})^2 k_2 \left(\frac{1}{i} \sum_{j=1}^{i} \nabla^2 L_{p,j}(\mathbf{x}) \right). \tag{19}$$

Finally, as with radiance estimation in Equation 16, we can expand the Laplacian of the progressive estimate as $\nabla^2 L_i(\mathbf{x}) = \frac{1}{i} \sum_{j=1}^{i} \nabla^2 L_{p,j}(\mathbf{x})$ using the recursion relation in Equation 15. This provides the following identity

$$\begin{aligned} &= \frac{1}{2} R_i(\mathbf{x})^2 k_2 \left(\frac{1}{i} \sum_{j=1}^{i} \nabla^2 L_{p,j}(\mathbf{x}) \right) \\ &= \frac{1}{2} R_i(\mathbf{x})^2 k_2 \nabla^2 L_i(\mathbf{x}) \\ &= B'_i \approx B_i, \end{aligned} \tag{20}$$

where we denoted our approximation of bias as B'_i. As a result, we can approximate the exact bias B_i just by computing a progressive estimate of $\nabla^2 L_i(\mathbf{x})$. We will describe more details for how to estimate $\nabla^2 L_i(\mathbf{x})$ in Section 5.

We tested our bias estimation for progressive photon mapping on a scene where we mimic a soft shadow boundary using an analytic cubic smoothstep function. We show a visualization of the resulting radiance function in Figure 2(a) while in Figure 2(b) we compare the actual bias to our estimated bias. In this scene, samples are generated using the cubic smoothstep function with importance sampling. Although the estimated bias does not exactly predict the actual bias, it captures two peaks of bias around edges of the smoothstep function as well as zero bias in the smooth region.

4.2 Variance Estimation

In standard kernel density estimation, variance is approximated as [Silverman 1986]:

$$V_i \approx \frac{1}{N_e^i R_i^2} k_3 L_i(\mathbf{x}), \tag{21}$$

where $k_3 = \int K(t)^2 \vec{dt}$. This approximation assumes that samples are independent and identically distributed (i.i.d.). Since this is true in the case of traditional photon mapping, this approximation can be used directly for photon mapping.

Unfortunately, this is not directly usable in progressive radiance estimation. Note that the samples in progressive radiance estimation are each of the radiance estimations using photons from *only* the j^{th} iteration, $L_{p,j}$. Since the radius reduction however depends on all previous photons, $L_{p,j}$ are *not statistically independent*. Furthermore, each $L_{p,j}$ gives us a biased estimation with a different radius R_j, thus $L_{p,j}$ are *not identically distributed*.

The key point is that photon tracing itself is statistically independent between each iteration in progressive photon mapping. The only dependence between samples is caused by the radius reduction. Therefore, instead of using samples ($L_{p,j}$) directly, we use bias-corrected samples $L_{p,j} - B_{p,j} \approx L_{p,j} - B'_j$ in order to estimate variance. The bias-corrected samples are expected to be closer to statistically independent since dependence on radius is removed by $B_{p,j}$, and identically distributed because $L_{p,j} - B_{p,j}$ is distributed according to the true radiance L.

Using $L_{p,j} - B_j$, we use the following standard procedure to estimate sample variance V_i at each iteration:

$$V_i \approx \frac{\sum_{j=1}^{i} {x_j}^2 - \left(\sum_{j=1}^{i} x_j \right)^2 / i}{i - 1} \tag{22}$$

where we use $x_j = L_{p,j} - B'_j$ as samples.

Note that $L_{p,j} - B'_j$ is not strictly unbiased or independent because B'_j is just an approximation of true bias. However, we found that the effect of this approximation is not statistically significant in practice through numerical experiments. Figure 3 shows that the autocorrelation of bias-corrected samples is almost 0 compared to the uncorrected samples. This indicates that the samples can be considered independent. Figure 3 furthermore shows that the bias correction makes the distribution of noise closer to the Gaussian distribution with mean 0, which is the distribution of ideal i.i.d. samples.

5 Kernel-based Progressive Photon Mapping

The previously described bias estimation method relies on the derivatives of radiance, $\nabla^2 L_i(\mathbf{x})$. To use this method, we must therefore estimate derivatives of radiance as well as radiance itself during rendering. In traditional photon mapping, derivatives can be easily estimated by just using derivatives of the kernel. However, estimation based on this method does not converge into the correct derivatives of radiance without using an infinite amount of memory (e.g. storing an infinite number of photons or using an infinite mesh tessellation).

In this section, we provide a new, generalized formulation of progressive radiance estimation which allows for arbitrary smooth kernels. We show that we can relax the "locally uniform density" assumption used in the original progressive photon mapping algorithm [Hachisuka et al. 2008], and that the kernel should be radially

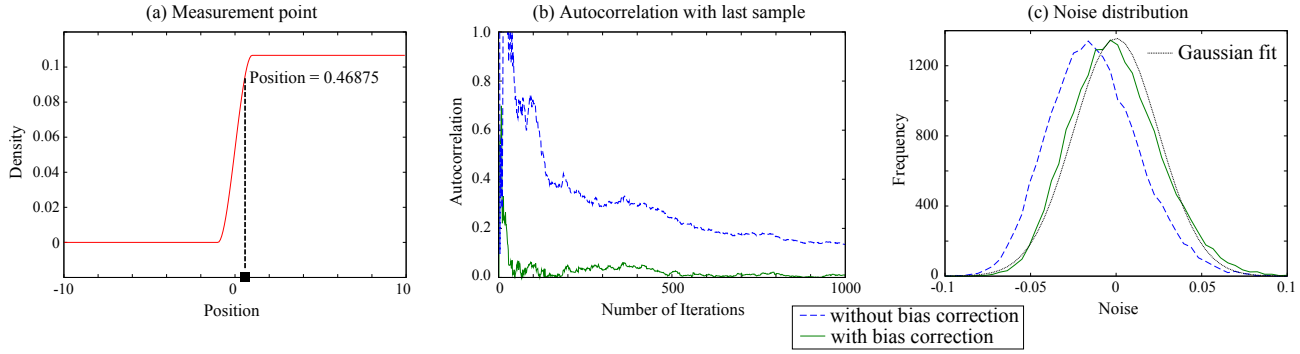

Figure 3: *Autocorrelation with one previous sample and noise distribution in the cubic ramp test scene. We measured autocorrelation and noise distribution on the point with largest bias shown in the graph (a). The graph (b) shows autocorrelation between one previous sample and current sample. The graph (c) shows the histogram of noise with 500000 independent runs. The Gaussian fit is using the sample variance of computed noise and mean* 0*.*

symmetric and C^n continuous at the boundary to be able to estimate nth order derivatives. This generalization makes estimation of derivatives straightforward and convergent, which in turn allows us to estimate bias more accurately.

5.1 Kernel-based Flux Correction

Kernel-based density estimation is a standard extension to traditional density estimation where the contribution of each sample in the density estimate is weighted by a, typically smooth, kernel K around the query location. In contrast to standard radiance estimation methods, simply replacing the constant kernel in the original progressive radiance estimate may not work. Specifically, since the radius in progressive radiance estimation changes, the accumulated flux needs to be corrected after radius reduction. To adapt kernel density estimation to progressive photon mapping, we need to derive an equation similar to Equation 7 (relating the accumulated flux before reducing the radius, τ'_{i+1}, with the accumulated flux after reducing the radius, τ_{i+1}) but in the presence of a weighting kernel K.

The accumulated flux before reducing the radius can be expressed in the continuous setting as

$$\tau'_{i+1} = \int_0^{2\pi}\int_0^{R_i} K\left(\frac{r}{R_i}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\; dr\; d\theta. \qquad (23)$$

We parametrized radiance over a circle using polar coordinates. Note that τ'_{i+1} is a function of $(\mathbf{x}, \vec{\omega})$, and R_i is a function of $\mathbf{x}$, but we drop this dependence in our notation for simplicity.

The accumulated flux with a reduced radius R_{i+1} is:

$$\tau_{i+1} = \int_0^{2\pi}\int_0^{R_{i+1}} K\left(\frac{r}{R_{i+1}}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\; dr\; d\theta. \qquad (24)$$

Given these two equations, we can express τ_{i+1} in terms of τ'_{i+1} as:

$$\tau_{i+1} = \tau'_{i+1} \frac{\int_0^{2\pi}\int_0^{R_{i+1}} K\left(\frac{r}{R_{i+1}}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\; dr\; d\theta}{\int_0^{2\pi}\int_0^{R_i} K\left(\frac{r}{R_i}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\; dr\; d\theta}. \qquad (25)$$

We can simplify this expression by performing a change of variable with $t = \frac{r}{R_i}$, $dr = R_i\; dt$, and analogously for R_{i+1}:

$$\begin{aligned}\tau_{i+1} &= \tau'_{i+1} \frac{\int_0^{2\pi}\int_0^1 K(t,\theta)\, L_r(tR_{i+1}, \theta, \vec{\omega})\, t\, {R_{i+1}}^2\; dt\; d\theta}{\int_0^{2\pi}\int_0^1 K(t,\theta)\, L_r(tR_i, \theta, \vec{\omega})\, t\, {R_i}^2\; dt\; d\theta}, \\ &= \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\int_0^{2\pi}\int_0^1 K(t,\theta)\, L_r(tR_{i+1}, \theta, \vec{\omega})\, t\; dt\; d\theta}{\int_0^{2\pi}\int_0^1 K(t,\theta)\, L_r(tR_i, \theta, \vec{\omega})\, t\; dt\; d\theta}.\end{aligned} \qquad (26)$$

Assuming that a kernel is radially symmetric $K(t, \theta) = K(t)$, the above expression can be further simplified by swapping the order of integration:

$$\tau_{i+1} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\int_0^1 K(t)\, t \int_0^{2\pi} L_r(tR_{i+1}, \theta, \vec{\omega})\; d\theta\; dt}{\int_0^1 K(t)\, t \int_0^{2\pi} L_r(tR_i, \theta, \vec{\omega})\; d\theta\; dt}. \qquad (27)$$

Finally, to simplify this expression further, we assume that the integrated radial distribution of radiance is locally constant such that

$$C = \int_0^{2\pi} L_r(tR_i, \theta, \vec{\omega})\; d\theta = \int_0^{2\pi} L_r(tR_{i+1}, \theta, \vec{\omega})\; d\theta, \qquad (28)$$

for some constant C. This assumption is similar, but not as strict as the "locally uniform density" assumption used in the original progressive photon mapping algorithm [Hachisuka et al. 2008]. Although the end result looks similar to the original progressive photon mapping, this distinction is crucial to use a smooth kernel function as well as its derivatives. Note also that the original assumption implies zero derivatives, which makes it impossible to use for the derivative estimation in our framework.

Using our new assumption, Equation 27 can be simplified to:

$$\tau_{i+1} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\cancel{\int_0^1 K(t)\, t\, C\; dt}}{\cancel{\int_0^1 K(t)\, t\, C\; dt}} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2}. \qquad (29)$$

This ends up being the same flux correction as in regular progressive photon mapping in Equation 7; however, it uses a weaker assumption and can be used with arbitrary radially symmetric kernels. Fortunately, it also means that the only change in terms of implementation is to weight each photon by a kernel function, thus it is straightforward to implement.

5.2 Progressive Radiance Derivative Estimation

The kernel-based estimation gives us an elegant way to estimate derivatives. Computing the derivatives simply involves replacing the

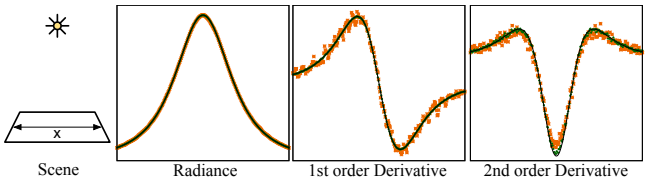

Figure 4: *Visualizations of radiance including 1st and 2nd order derivatives for a simple scene containing a Lambertian plane and a point light source. We show the estimated values using progressive photon mapping with 10K (orange) and 1M (green) stored photons, and compare to the analytic solutions (black). As more photons are used, the results converge to the analytic solutions.*

kernel with a corresponding derivative kernel. In the following, we describe how to compute the first-order derivative for simplicity.

The first-order derivative of the radiance estimate can be computed directly as:

$$\nabla L(\mathbf{x}, \vec{\omega}) \approx \nabla\left(\frac{\tau_i}{k_1 R_i^2}\right) = \frac{\nabla\tau_i}{k_1 R_i^2}, \tag{30}$$

where the derivative of the accumulated flux is:

$$\begin{aligned}\nabla\tau_i &= \frac{1}{N_i^e}\,\nabla\left(\sum_{p=1}^{N_i} K(t)\, f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p)\, \Phi_p(\mathbf{x}_p, \vec{\omega}_p)\right) \\ &= \frac{1}{N_i^e}\sum_{p=1}^{N_i}(\nabla K)\, f_r\, \Phi_p + K\,(\nabla f_r)\,\Phi_p \\ &= \frac{1}{N_i^e}\sum_{p=1}^{N_i}(\nabla K)\, f_r\, \Phi_p.\end{aligned} \tag{31}$$

We always use the gradient operator ∇ to denote a gradient with respect to the position parameter $\mathbf{x}$, and the $\nabla\Phi_p$ term from the product rule is omitted above because $\nabla\Phi_p = 0$. The ∇f_r term is also omitted since the BRDF within the radius is locally constant (i.e., BRDF is defined by the measurement point) thus $\nabla f_r = 0$.

Higher-order derivatives can be derived in exactly the same way by simply taking higher-order derivatives of the kernel function. We use the second order derivatives $\nabla^2 L_i(\mathbf{x})$ in order to estimate the bias in Equation 17.

In order to estimate derivatives in a progressive manner, we need to express the accumulated flux derivatives after radius reduction, $\nabla\tau_{i+1}$, in terms of the accumulated value before radius reduction, $\nabla\tau'_{i+1}$. This relation can be established by differentiating Equation 29:

$$\nabla\tau_{i+1} = \nabla\left(\tau'_{i+1}\frac{R_i^2}{R_{i+1}^2}\right) = (\nabla\tau'_{i+1})\frac{R_i^2}{R_{i+1}^2}. \tag{32}$$

Hence, any order flux derivative is accumulated using the ratio of the squared radii, just like accumulated flux. Again, the only change in terms of implementation is replacing the kernel with derivatives of the kernel.

One condition on the kernel is that derivatives have to be finite in order for this computation to be meaningful. For example, if we use a Gaussian by cutting off the kernel to a finite radius, this gives us infinite derivatives along the edge of the kernel, which is not useful. The same thing happens with a Epanechnikov kernel, which is a popular choice in standard density estimation. We therefore found that the kernel has to be C^n continuous with 0 at the boundary for computing nth order derivatives (i.e., the nth derivative of the kernel must go to 0 at the edge). We provide an example of a kernel function that satisfies this condition in Appendix A.

In order to investigate the accuracy of the progressive kernel radiance estimate and the radiance derivative estimate, we used a simple scene with a point light source, where all the values can be computed analytically in Figure 4. The estimated radiance values match the analytical solution very closely with only 10K stored photons. Although the radiance derivative estimates are noisy with 10K stored photons, they are converging to the analytical solution with 1M stored photons.

6 Results

We have implemented our error estimation on top of an implementation of the progressive photon mapping algorithm. All results were computed on a 2.4GHz Intel Core 2 Q6600 using one core. In all examples, we used the parameter $\alpha = 0.8$ in the progressive photon mapping algorithm.

Figure 5 is an analytic investigation of our error estimation. We used the cubic ramp scene described in Figure 2 with 640000 samples and an initial radius of 1.5. The user-specified confidence is $1 - \beta = 0.95$. The graph (b) shows that the estimated error is in fact above the actual error in almost all locations as specified. We further investigated the influence of bias estimation and noise bound estimation by plotting each of these components separately in the graph (c). Note that our framework does not estimate bounds of bias, but bias itself. This graph shows that the bias estimation captures error due to bias around the slope of the function, whereas the noise bound estimation captures error due to sampling noise around the flat region of the function.

We tested our framework on three example scenes. The reference renderings are shown in Figure 8. Each iteration uses 15k emitted photons and the confidence was specified at 90%. The first scene is a Cornell box with a glass sphere and a metal sphere. Here noise dominates the error because the scene is composed of mostly large flat regions of fairly uniform illumination. The second scene consists of three flashlights models. We modeled the filament, lens and reflector of the flashlight in order to simulate a realistic lighting pattern. This scene is expected to have larger bias due to sharp caustics from the flashlights. The third scene is a room with lighting fixtures. Each lighting fixture consists of a filament, a glass casing, and reflectors which replicates light transport of real-world lighting fixtures. This scene contains both sharp illumination features due to caustics as well as flat and continuous illumination, which is a realistic mixture of bias and noise in error. Note that both the flashlights scene and the room scene have light paths which are difficult for unbiased rendering methods to handle (such as an SDS path from a very small light source [Hachisuka et al. 2008]).

Figure 8 shows behavior of the actual error and the estimated error for the test scenes. We picked three points to investigate the actual error and the estimated error as shown in the leftmost column of Figure 8. One might notice that our estimated error does not bound actual error at some points, such as the point A in the room scene. This is indeed the expected behavior because our error estimation is motivated by stochastic error bounds. Stochastic error bounds, by definition, bound the actual error with some probability, so even if our error estimate is perfect, some points where the actual error is higher than the estimation are to be expected. In practice, since bias estimation is an approximation, it may also cause extra overestimation or underestimation of the actual error in addition to what is expected from exact stochastic error bounds.

What is important in practice is whether the estimated error respect

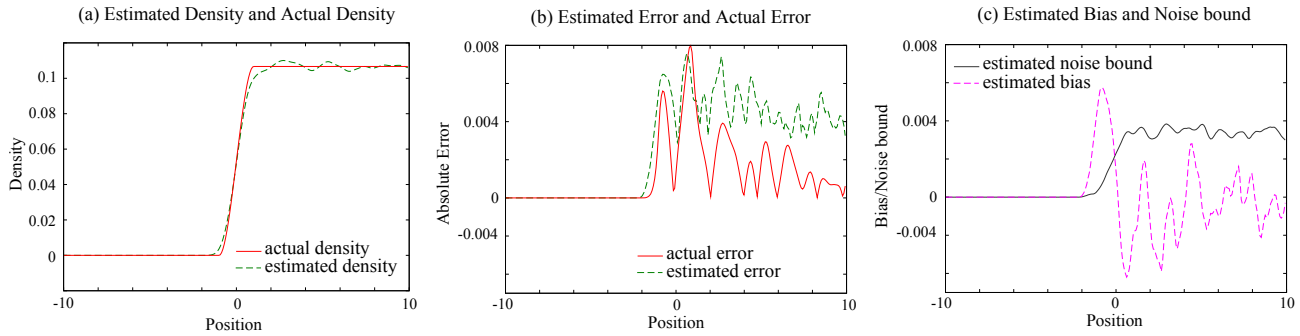

Figure 5: *Error estimation on the cubic ramp scene. The graphs show (a) estimated density and actual density, (b) estimated error (95%) and actual error, and (c) estimated bias and estimated noise bound with 640000 samples. The estimated bias dominates the error estimation around the slope, while the estimated noise bound dominates the error estimation around the flat region.*

the user-specified confidence. In order to check this, we computed the actual confidence on the test scenes in Figure 7. The ideal behavior is that the actual confidence stays at the user-specified confidence all the time. We calculate the actual confidence as the percentage of all error estimates that are above the true errors obtained using the reference image. We tested 50% and 90% as the user-specified confidence. Although the calcuated confidence is not exactly the same as the specified confidence, the deviation from the user-specified confidence is within 5% across many iterations for both 50% and 90% in difference scenes. The yellow and blue images in Figure 1 visualize whether estimated error is above error per pixel (yellow: bounded, blue: not bounded). Note also that error estimation itself becomes more reliable as we add more samples because the magnitude of the error estimate decreases.

Figure 6 shows the ratio of estimated bias to estimated noise bound using 15M photons. The estimated error captures noise in flat regions as well as bias due to sharp features in practical scenes. The bias-to-noise ratio images demonstrate the importance of each of these components to our overall error estimate. The resulting images show bias estimation dominates the error estimation around sharp features (shown as red) and the noise bound estimation dominates the error estimation within flat regions (shown as green).

6.1 Rendering Termination

Predicting a sufficient total number of photons to obtain a desired quality for a particular scene is difficult. Unfortunately, this is currently set by tedious trial-and-error by the user. As an example application of our error estimation, we propose an automatic way to stop the rendering process without specifying the total number of photons or the total render time. Instead, we stop the rendering process when the average estimated error over the image is below a user specified threshold. We believe that this provides a more meaningful termination criterion, which allows the user to focus on the quality of the resulting image and not abstract parameters involved in the rendering algorithm.

At each iteration, we simply compute the average estimated error

Scene	Triangles	$E^{\text{thr}} = 0.5$	$E^{\text{thr}} = 0.25$	$E^{\text{thr}} = 0.125$	$E^{\text{thr}} = 0.0625$
Cornell Box	9666	6	55	340	1969
Flashlights	38118	2	36	229	1410
Room	181168	13	93	570	3465

Table 2: *The number of triangles in each scene and the number of iterations to reach the specified error threshold. Note that processing time for each iteration is different for each scene.*

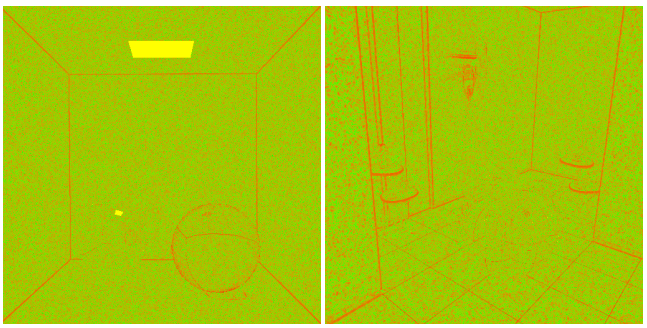

Figure 6: *The ratio of the estimated bias and the estimated noise bound in error estimation with 15M emitted photons. The red pixel indicates the bias is dominant and the green pixel indicates the noise is dominant in the estimated error. The result for the flashlights scene is in Figure 1.*

over all the measurement points:

$$E^{\text{ave}} = \frac{1}{N_m} \sum_{j=1}^{N_m} \frac{E_i(\mathbf{x}_j) + |B'_i(\mathbf{x}_j)|}{L_i(\mathbf{x}_j)}, \tag{33}$$

where N_m is the number of measurement points. The rendering process stops when $E^{\text{ave}} \leq E^{\text{thr}}$ for a given error threshold E^{thr}. Note that we could even use our error estimation to individually terminate the computation of *each* measurement point since the error is estimated per measurement point. Here, we use average error as a proof of concept that shows our error estimation is useful for a practical application. Since we are interested in the average error, not per-pixel error, we choose rather lower 50% confidence in this application in order to get tighter error estimation.

Figure 9(a) shows a sequence of color-coded images of actual error with different user-specified thresholds. We plot the graphs of actual average relative error and the average of estimated error in Figure 9(b). Although we used 50% confidence to terminate the rendering process, we also show the resulting estimated average error using 90% confidence for reference. The average of our error estimate using 50% confidence successfully predicts, without any user input other than the threshold, that different scenes need a different number of iterations to obtain sufficient quality.

Table 2 shows the number of iterations used for achieving the specified threshold. Note that, even though the flashlight scene has more triangles than the Cornell box scene, the number of iterations to achieve the same threshold is less than that of the Cornell box.

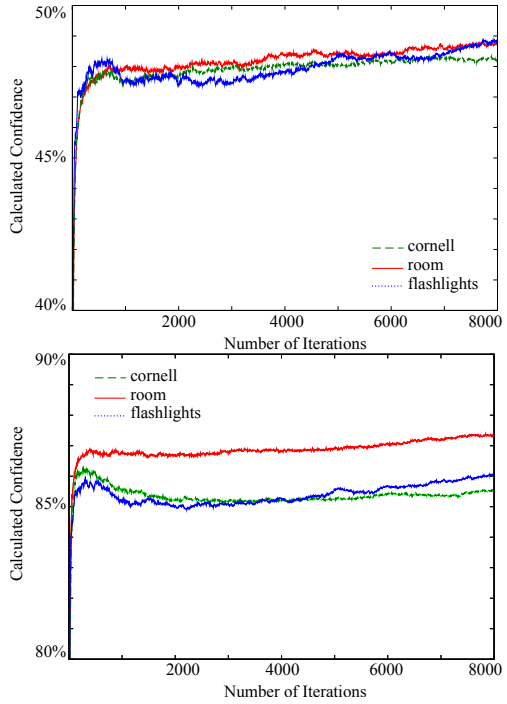

Figure 7: *Calculated confidence of the estimated error in different scenes. The confidence is calculated as the percentage of estimated error that is actually larger than actual error. The graph on the top uses 50% and the graph on the bottom uses 90% as the user-specified confidence.*

The actual average relative errors achieved with $E^{\text{thr}} = 0.0625$ are 0.0465, 0.0448, 0.0437 for the Cornell box scene, the flashlight scene, and the room scene respectively. Although the statement that different scenes need different number of samples sound trivial, we do not know how many samples are needed unless we actually render an image. We believe that even this simple application will shed light on theoretical analysis of the convergence behavior of progressive density estimation, which is still unknown.

Since our error estimation framework does not need additional photon tracing or range query, runtime overhead due to the error estimation is very small. In our implementation, each pass including the error estimation took 681ms, 1291ms, and 720ms on average for the Cornell box scene, the flashlight scene, and the room scene respectively, and the overhead due to the error estimation are 2.2ms, 5.9ms, and 3.3ms which are all less than 1% of the rendering time without error estimation.

7 Discussion and Limitation

Our error estimation framework depends on the number of assumptions. We discuss some issues due to those assumptions we made in this section to help readers understand the limitations of our method. In general, our experience along with previous work concludes that making assumptions or approximations is probably inevitable and providing theoretical guarantees that error estimation works is challenging.

First, the derivations in Equation 10 are exact only if we know the true bias and variance. Since we approximated bias and variance in our error estimation with a finite number of samples, the resulting error estimation can be arbitrary far away from the exact error. This might seem to be a critical flaw as an error estimation framework, however, variance-based error estimation in unbiased methods have also made the approximation of assuming normal distribution of error, which is not usually the case. Even with such an approximation, variance-based error estimation has been used in many practical applications as we mentioned in Section 2. We therefore believe that using approximations does not make our error estimation completely useless in practice. Notice however that we do not claim our framework works well in any possible scene setting.

Second, related to the above issue, our bias estimation is not a consistent estimator of the true bias (i.e., does not give us the true bias in the limit). This in turn affects variance estimation, which assumes i.i.d. samples after bias correction. We approximate bias using the Laplacian of radiance (Equation 17) and we introduced one additional approximation between Equations 18 and 19 in order to make bias estimation practical. The approximation by Laplacian can cause significant error in bias estimation (either underestimation or overestimation) if the radiance distribution contains significant higher order derivatives (higher or equal to 4th order) and the radius is relatively large. We believe accurate bias estimation, in general, is an interesting direction of future work, especially since it can be useful for bias reduction, and not only error estimation.

Finally, our application of error estimation for automatic rendering termination may not be well-suited for some usage. For example, if we want to avoid generating redundant samples by terminating the rendering process after the desired rendering error is achieved, our application will result in oversampling due to slight overestimation of the rendering error. In such cases, more accurate error estimation is needed to avoid oversampling. Although our overestimation is generally around 1.4 times throughout many iterations in our experiments as in Section 6.1, this could mean taking at least $1.4^2 = 1.96$ times more samples than absolutely necessary, according to $O(i^{-\frac{1}{2}})$ error convergence rate of Monte Carlo integration. Not generating redundant samples would require the estimated error to be equal to the exact error, which is challenging to accomplish in general.

8 Conclusion

We have presented an error estimation framework for progressive photon mapping, which is robust to a wide range of light paths. Based on the derivations of stochastic error bounds, we characterized error using estimated bias and an estimated noise bound. We estimate bias using derivatives of radiance, and for this purpose we have extended the progressive radiance estimate to work with arbitrary smooth kernels. Using a kernel-based approach we can compute the derivatives of radiance by using the derivatives of the kernel. We have shown that this bias estimate can be used to construct a variance estimator for the progressive radiance estimate. The estimated error captures error due to both bias and noise under complex illumination given user-defined confidence. We have demonstrated that our error estimate can be used to automatically stop the rendering process. We believe that our work is the first step towards answering the important question: “How many photons are needed to render this scene?”.

There are several possible areas of future work that we would like to explore. Given an error estimate, we can try to accelerate progressive photon mapping using adaptive sampling by tracing photons into the parts of the scene where the error is large. It would be interesting to enhance the error estimate with a perceptual metric such as Ramasubramanian et al. [1999]. We would also like to investigate if the error estimate can be extended to include stochastic progressive photon mapping [Hachisuka and Jensen 2009], which would need estimation of average bias over an unknown region.

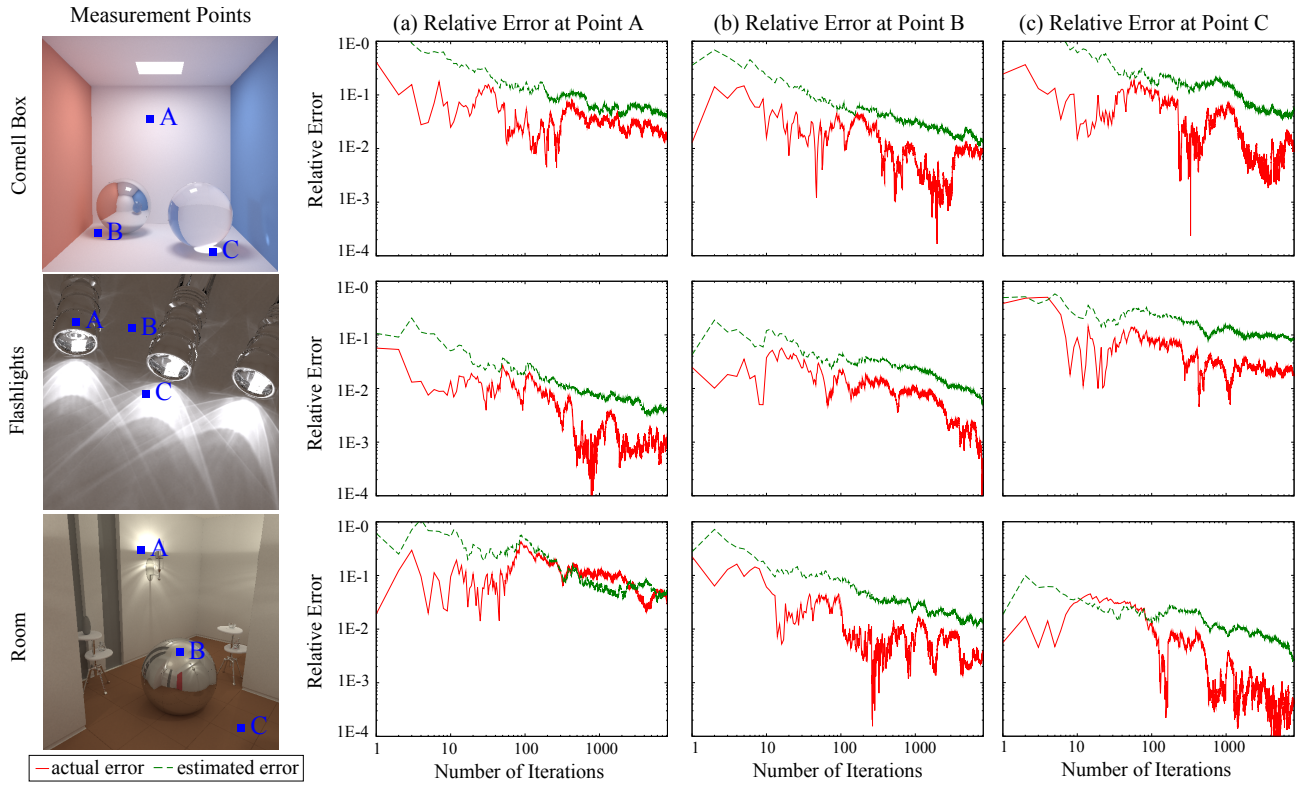

Figure 8: *Error estimation on test scenes. Each row shows the reference rendering (left), as well as the actual error (red) and estimated error (green) at the three points (a,b,c) shown in the reference images. The specified confidence is 90%, and each iteration uses 15k photons.*

Acknowledgments

We would like to thank Youichi Kimura (Studio Azurite) for providing us the flashlight model and the room model. We would also like to thank anonymous reviewers for suggestions on improving the presentation of the paper. The first author was supported by the 2008-2009 ATI Fellowship Program.

References

FIORIO, C. V. 2004. Confidence intervals for kernel density estimation. *Stata Journal 4*, 2, 168–179(12).

GEERTSEMA, J. C. 1970. Sequential confidence intervals based on rank tests. *The Annals of Mathematical Statistics 41*, 3, 1016–1026.

GLASSNER, A. S. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann.

HACHISUKA, T., AND JENSEN, H. W. 2009. Stochastic progressive photon mapping. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, ACM, New York, NY, USA, 1–8.

HACHISUKA, T., OGAKI, S., AND JENSEN, H. W. 2008. Progressive photon mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008) 27*, 5, Article 130.

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

HAVRAN, V., BITTNER, J., HERZOG, R., AND SEIDEL, H.-P. 2005. Ray maps for global illumination. In *Eurographics Symposium on Rendering*, 43–54.

HERZOG, R., HAVRAN, V., KINUWAKI, S., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2007. Global illumination using photon ray splatting. *Computer Graphics Forum 26*, 3, 503–513.

HEY, H., AND PURGATHOFER, W. 2002. Advanced radiance estimation for photon map global illumination. *Computer Graphics Forum 21*, 3.

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. Irradiance Gradients in the Presence of Participating Media and Occlusions. *Computer Graphics Forum (Proceedings of EGSR 2008) 27*, 4, 1087–1096(10).

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA.

KAJIYA, J. T. 1986. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, D. C. Evans and R. J. Athay, Eds., vol. 20, 143–150.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bidirectional path tracing. In *Proceedings of CompuGraphics*, 95–104.

LEE, M. E., REDNER, R. A., AND USELTON, S. P. 1985. Statistically optimized sampling for distributed ray tracing. *SIGGRAPH Comput. Graph. 19*, 3, 61–68.

MYSZKOWSKI, K. 1997. Lighting reconstruction using fast and adaptive density estimation techniques. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, Springer-Verlag, London, UK, 251–262.

PERLIN, K. 2002. Improving noise. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 681–682.

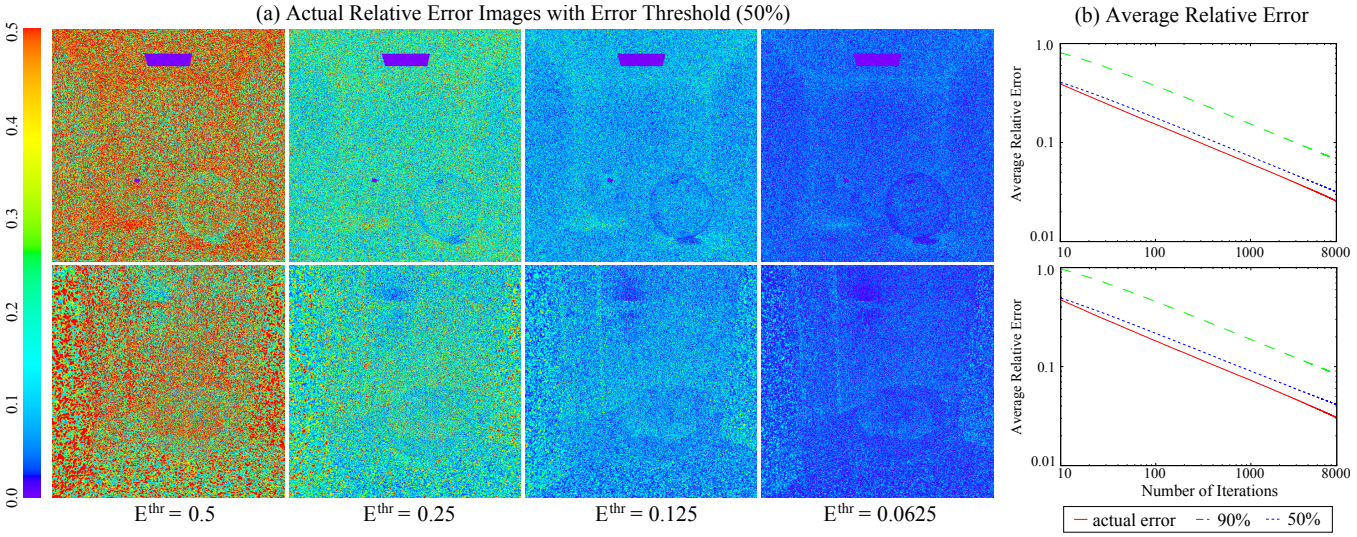

Figure 9: *Color-coded relative error images with specified average estimated error. The confidence used is $1 - \beta = 50\%$ for estimating the average error. The rendering process of each column is terminated after the average estimated relative error estimate is smaller than the threshold which is shown below. The graphs on the right show actual average relative error (red) and the estimated average relative errors with different confidence values (green: 90% confidence, blue 50% confidence). The results for the flashlights scene are in Figure 1.*

PURGATHOFER, W. 1987. A statistical method for adaptive stochastic sampling. *Computer & Graphics*, 11. Pergamon Press, New York.

RAMAMOORTHI, R., MAHAJAN, D., AND BELHUMEUR, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2.

RAMASUBRAMANIAN, M., PATTANAIK, S. N., AND GREENBERG, D. P. 1999. A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 73–82.

SCHREGLE, R. 2003. Bias compensation for photon maps. *Comput. Graph. Forum 22*, 4, 729–742.

SILVERMAN, B. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York, NY.

TAMSTORF, R., AND JENSEN, H. W. 1997. Adaptive smpling and bias estimation in path tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, Springer-Verlag, London, UK, 285–296.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH 95 Conference Proceedings*, 419–428.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. 24*, 3, 1098–1107.

WALTER, B. 1998. *Density Estimation Techniques for Global Illumination*. PhD thesis, Ithaca, NY, USA.

WARD, G. J., AND HECKBERT, P. S. 1992. Irradiance gradients. *Third Eurographics Workshop on Rendering*, 8598.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 85–92.

A Kernels

In order to compute the first and second order derivatives, the kernel function must be at least twice differentiable. We use a sixth-order smooth kernel [Perlin 2002] for the progressive kernel radiance estimation. The kernel is defined as:

$$K(t) = 1 - 6t^5 + 15t^4 - 10t^3. \tag{34}$$

Since K is a polynomial, its derivatives can easily be computed as:

$$K'(t) = -30t^4 + 60t^3 - 30t^2, \quad K''(t) = -120t^3 + 180t^2 - 60t.$$

The normalization constants are $k_1 = \frac{2\pi}{7}$, and $k_2 = \frac{10\pi}{168}$.