Stochastic Progressive Photon Mapping

Toshiya Hachisuka Henrik Wann Jensen

UC San Diego

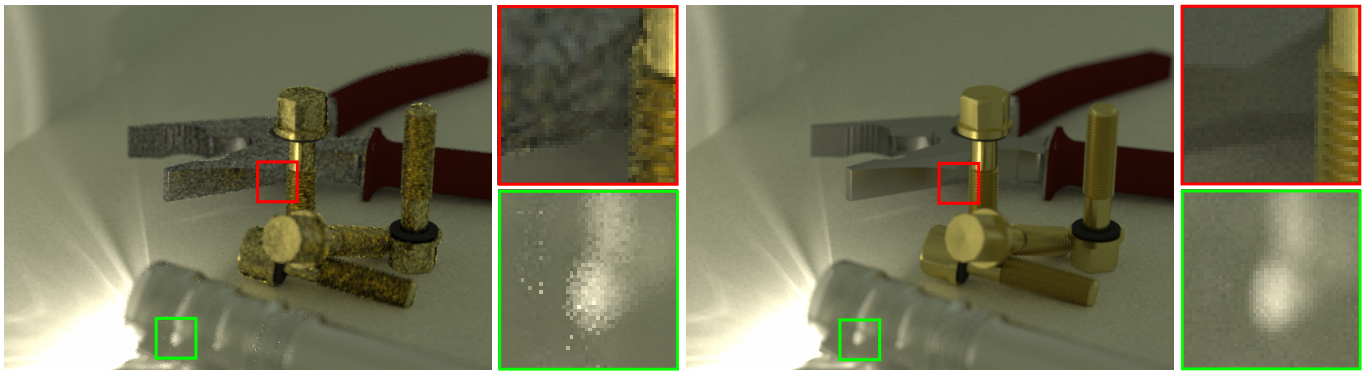


Figure 1: *Tools with a flashlight. The scene is illuminated by caustics from the flashlight, which cause SDS paths on the flashlight and highly glossy reflections of caustics on the bolts and plier. The flashlight and the plier are out of focus. Using the same rendering time, our method (right) robustly renders the combination of the complex illumination setting and the distributed ray tracing effects where progressive photon mapping is inefficient (left).*

Abstract

This paper presents a simple extension of progressive photon mapping for simulating global illumination with effects such as depth-of-field, motion blur, and glossy reflections. Progressive photon mapping is a robust global illumination algorithm that can handle complex illumination settings including specular-diffuse-specular paths. The algorithm can compute the correct radiance value at a point in the limit. However, progressive photon mapping is not effective at rendering distributed ray tracing effects, such as depth-of-field, that requires multiple pixel samples in order to compute the correct average radiance value over a region. In this paper, we introduce a new formulation of progressive photon mapping, called stochastic progressive photon mapping, which makes it possible to compute the correct average radiance value for a region. The key idea is to use shared photon statistics within the region rather than isolated photon statistics at a point. The algorithm is easy to implement, and our results demonstrate how it efficiently handles scenes with distributed ray tracing effects, while maintaining the robustness of progressive photon mapping in scenes with complex lighting.

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

ACM Reference Format
Hachisuka, T., Jensen, H. 2009. Stochastic Progressive Photon Mapping. *ACM Trans. Graph. 28*, 5, Article 141 (December 2009), 8 pages. DOI = 10.1145/1618452.1618487 http://doi.acm.org/10.1145/1618452.1618487.

1 Introduction

Efficiently computing global illumination is an active areas of research in computer graphics. The types of lighting can vary significantly in different scenes, and it is important to develop algorithms that can handle this variation robustly.

Global illumination algorithms solve the rendering equation introduced by Kajiya [1986]. Unbiased Monte Carlo methods have been a popular approach for computing global illumination without any approximations in the past few decades [Dutré et al. 2006]. However, unbiased methods are not robust under all illumination settings. There are certain light paths that are problematic. For example, path tracing [Kajiya 1986] works well for a scene with diffuse materials, however, it cannot efficiently handle caustics from a small light source.

Hachisuka et al. [2008] observed out that specular-diffuse-specular paths (SDS paths in the light path notation) are particularly problematic for the existing unbiased methods. An example of an SDS path is a light path due to the combination of specular materials and a light bulb. Photon mapping [Jensen 1996] is a biased method which is robust in the presence of SDS paths. However, the results suffer from bias, which appears as low frequency noise in the rendered images. Moreover, computing the correct solutions requires storing an infinite number of photons in the limit. Progressive photon mapping [Hachisuka et al. 2008] solves this issue by using progressive refinement, and makes it possible to compute a correct solutions without storing any photons. Moreover, the method retains the robustness of photon mapping.

Although each radiance estimate in progressive photon mapping converges to the correct radiance, the algorithm is restricted to computing the correct radiance value at a *point*. This property limits the applications of progressive photon mapping because we often need to compute the correct average radiance value over a *region*. For example, anti-aliasing in ray tracing requires the average radiance value for a pixel footprint. Depth-of-field is another example where each pixel value is the average radiance value for the part of a scene that is visible through the lens. In general, all the effects that can

be achieved by distributed ray tracing [Cook et al. 1984] require computing the average radiance over some domain.

In this paper, we present a new formulation of progressive photon mapping that enables computing the correct average radiance value over a region. Our formulation requires a simple algorithmic modification, which consists of adding a distributed ray tracing pass after each photon pass in progressive photon mapping. The main contribution is this new formulation that allows simple, yet effective improvement of the robustness of the progressive photon mapping. We show that our modification allows us to render scenes with distributed ray tracing effects in combination with complex illumination scenarios.

2 Related Work

Global illumination is simulated by solving the rendering equation [Kajiya 1986]. The most popular approaches for solving the rendering equation are based on Monte Carlo sampling, and these methods can further be classified as either unbiased methods or biased methods. Path tracing [Kajiya 1986] is one of the unbiased methods that solves the rendering equation by constructing light paths starting from the camera. Since a light path starting from the camera has to probabilistically reach a light source in order to produce caustics, path tracing is not robust in scenes with a small light source and specular materials.

Bidirectional path tracing [Lafortune and Willems 1993; Veach and Guibas 1995] avoids this issue by constructing a light path starting from both the camera and light sources at the same time. If a path from a light source hits a diffuse surface, bidirectional path tracing directly connects the hit point with the camera in order to compute the contribution to the image, which efficiently handles caustics from a small light source. However, bidirectional path tracing still fails to render specular reflections/refractions of caustics. This is because directly connecting a point on a specular surface to the camera always results in zero contribution to the image.

The application of Markov Chain Monte Carlo have been shown to improve the efficiency of the path construction [Veach and Guibas 1997; Cline et al. 2005]. The key observation is that a path similar to the existing path with a large contribution tends to have a large contribution as well. A new light path is constructed from a mutation (perturbation) of the existing path, thus forming Markov Chain of paths. Unfortunately, this class of methods still fails to render specular reflections/refractions of caustics. The reason is that even a small mutation from the existing path with specular reflections/refractions often result in a path with zero contributions to the image.

Photon mapping [Jensen 1996] is a biased two-pass method that solves the rendering equation by estimating the density of photons. In the first pass, photons are traced from light sources and the resulting hit points on non-specular materials are stored as a photon map. In the second pass, rays are traced from the camera until they hit a non-specular material. The radiance value at the camera-ray hit point is then computed using density estimation. Since photon mapping loosely connects paths from light sources with paths from the camera by means of density estimation, it is robust in the presence of specular reflections/refractions of caustics. One critical issue is that it is necessary to store an infinite number of photons in order to compute the correct solution to the rendering equation. In other words, the accuracy of photon mapping is both memory and computationally bounded, whereas accuracy of unbiased methods is only computationally bounded.

Progressive photon mapping [Hachisuka et al. 2008] removed the memory bound of photon mapping, which makes the results converge to the correct solutions (i.e., bias goes to zero in the limit). The key idea is using progressive refinement of photon statistics at a point where the radiance value is computed. Those points are generated by the ray tracing similar to the second pass of photon mapping. The progressive refinement of statistics is achieved by a new progressive radiance estimate, which uses a new progressive density estimation technique. Progressive photon mapping ensures convergence to the correct radiance values with bounded memory consumption, and still retains the robustness of photon mapping. Our work is an extension of progressive photon mapping that improves its robustness to an even wider class of scene settings.

The problem we are dealing with in this paper is the computation of the average radiance value over an unknown region (i.e., unknown before computation). Such a problem arises when distributed ray tracing [Cook et al. 1984] is used for adding depth-of-field, motion blur, and glossy reflections/refractions. Unbiased methods can include these effects without changing the algorithms; however, this class of methods is not robust to complex illumination settings. In this paper, we extend progressive photon mapping in order to develop a new rendering algorithm that is robust to the combination of complex illumination settings and distributed ray tracing effects.

Computing the average radiance (density) over an unknown region is not a typical problem setting in density estimation methods outside graphics. We are not aware of existing work in density estimation literatures outside graphics (refer to [Silverman 1986; Wasserman 2006] for example). In computer graphics, there are few related methods that extend the radiance estimation for computing average radiance values. Time dependent photon mapping [Cammarano and Jensen 2002] computes the average radiance value over time by extending the space of photon mapping into 4D (3D position and time). The beam radiance estimate [Jarosz et al. 2008] computes the integration (weighted average of the radiance values) on a line of sight in order to accelerate rendering of participating media. These methods still store photons in a photon map similar to the standard photon mapping, thus suffering from unbounded memory consumption for the correct solutions. On the other hand, our method enables to compute the correct average radiance values with bounded memory consumption.

3 Overview

3.1 Progressive Photon Mapping

Progressive photon mapping [Hachisuka et al. 2008] is a multi-pass method that solves the rendering equation by accumulating statistics of photons. The initial eye pass traces rays from the camera and stores all the non-specular hit points. The following photon passes trace photons from light sources and update statistics on the hit points using those photons. The photon statistics include position of a hit point $\vec{x}$, accumulated (unnormalized) flux times BRDF $\tau_i(\vec{x},\vec{\omega})$, search radius $R_i(\vec{x})$, and the local accumulated photon count within the radius $N_i(\vec{x})$. Here, i is the number of photon tracing passes so far.

At the i-th pass, the radiance value at the position $\vec{x}$ toward the direction $\vec{\omega}$ is estimated as:

$$L(\vec{x},\vec{\omega}) \approx \frac{\tau_i(\vec{x},\vec{\omega})}{N_e(i)\pi R_i(\vec{x})^2}, \tag{1}$$

where $N_e(i)$ is the number of emitted photons after i passes and usually $N_e(i) \propto i$ (i.e., the number of emitted photons per pass is fixed). The photon statistics are updated with a new set of photons at each photon tracing pass. If $M_i(\vec{x})$ photons are found within the search radius $R_i(\vec{x})$ during the pass i, the progressive radiance

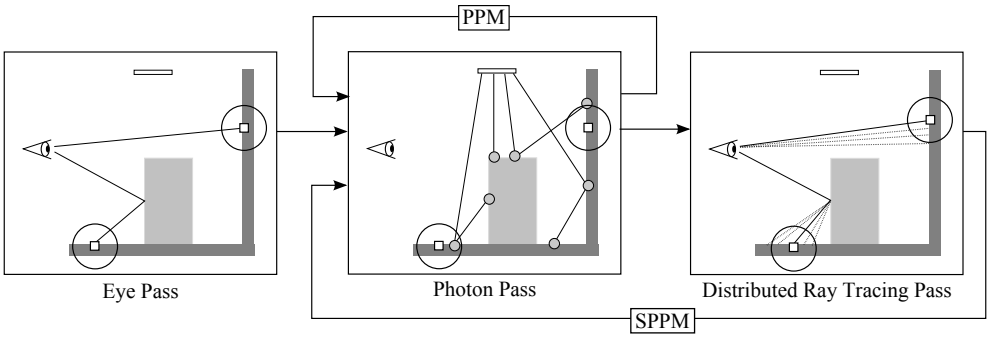


Figure 2: *Difference between the algorithms of progressive photon mapping (PPM) and stochastic progressive photon mapping (SPPM). In order to compute the average radiance values, SPPM adds a new distributed ray tracing pass after each photon tracing pass. The photon tracing algorithm itself stays the same, but PPM uses a fixed set of hit points (shown as squares), whereas SPPM uses randomly generated hit points by the distributed ray tracing pass.*

estimate updates the statistics as:

$$N_{i+1}(\vec{x}) = N_i(\vec{x}) + \alpha M_i(\vec{x}) \tag{2}$$

$$R_{i+1}(\vec{x}) = R_i(\vec{x}) \sqrt{\frac{N_i(\vec{x}) + \alpha M_i(\vec{x})}{N_i(\vec{x}) + M_i(\vec{x})}} \tag{3}$$

$$\Phi_i(\vec{x}, \vec{\omega}) = \sum_{p=1}^{M_i(\vec{x})} f_r(\vec{x}, \vec{\omega}, \vec{\omega}_p) \Phi_p(\vec{x}_p, \vec{\omega}_p) \tag{4}$$

$$\tau_{i+1}(\vec{x}, \vec{\omega}) = (\tau_i(\vec{x}, \vec{\omega}) + \Phi_i(\vec{x}, \vec{\omega})) \frac{R_{i+1}(\vec{x})^2}{R_i(\vec{x})^2}, \tag{5}$$

where $\alpha \in (0, 1)$ is a user-defined parameter, f_r is the BRDF, $\Phi_p(\vec{x}_p, \vec{\omega}_p)$ is the flux of photon p, and $\vec{\omega}_p$ is the incoming direction of photon p. The progressive radiance estimate ensures that the local photon count increases and the search radius decreases monotonically at the same time. Since this updating procedure satisfies the conditions of consistency [Silverman 1986], a radiance estimate converges to the correct radiance value with an infinite number of photon tracing passes:

$$L(\vec{x}, \vec{\omega}) = \lim_{i \to \infty} \frac{\tau_i(\vec{x}, \vec{\omega})}{N_e(i) \pi R_i(\vec{x})^2}. \tag{6}$$

3.2 Stochastic Progressive Photon Mapping

In this paper, we propose a new formulation of the progressive radiance estimate, called *stochastic progressive radiance estimate*, which can compute the correct average radiance value over a region. The motivation is that we need to compute the average radiance value over a region in order to render distributed ray tracing effects. For example, motion blur requires computing the average radiance value over a visible part of a scene for a given shutter time, and depth-of-field needs the average radiance value over a part of scene that is visible through a lens. As we have seen in Equation 6, the original progressive radiance estimate is restricted to computing the correct radiance value at a point $\vec{x}$.

Our idea is to use shared statistics over a region that we would like to compute the average radiance value for. Using the shared statistics, the stochastic progressive radiance estimate approximates the average radiance value $L(S, \vec{\omega})$ over the region S as:

$$L(S, \vec{\omega}) \approx \frac{\tau_i(S, \vec{\omega})}{N_e(i) \pi R_i(S)^2}, \tag{7}$$

where i is the number of photon passes as before, $\tau_i(S, \vec{w})$ is the shared accumulated flux over the region S, and $R_i(S)$ is the shared search radius. The updating procedure of the shared statistics is:

$$N_{i+1}(S) = N_i(S) + \alpha M_i(\vec{x}_i) \tag{8}$$

$$R_{i+1}(S) = R_i(S) \sqrt{\frac{N_i(S) + \alpha M_i(\vec{x}_i)}{N_i(S) + M_i(\vec{x}_i)}} \tag{9}$$

$$\Phi_i(\vec{x}_i, \vec{\omega}) = \sum_{p=1}^{M_i(\vec{x}_i)} f_r(\vec{x}_i, \vec{\omega}, \vec{\omega}_p) \Phi_p(\vec{x}_p, \vec{\omega}_p) \tag{10}$$

$$\tau_{i+1}(S, \vec{\omega}) = (\tau_i(S, \vec{\omega}) + \Phi_i(\vec{x}_i, \vec{\omega})) \frac{R_{i+1}(S)^2}{R_i(S)^2}, \tag{11}$$

where $\vec{x_i}$ is a randomly generated position within S and $N_i(S)$ is the shared local photon count. Note that the updating procedure is the same as before, except that our formulation uses a randomly generated position $\vec{x}_i$ in S. The next section describes why this change allows to estimate the average radiance value over S. The only algorithmic change from the original progressive photon mapping is that each hit point is randomly generated within the region S by distributed ray tracing after each photon pass. Figure 2 summarizes the difference between the algorithms of progressive photon mapping and our algorithm. Even though the modification is simple, the stochastic radiance estimate converges to the correct average radiance over S for $i \to \infty$ without the knowledge of S in advance:

$$L(S, \vec{\omega}) = \lim_{i \to \infty} \frac{\tau_i(S, \vec{\omega})}{N_e(i) \pi R_i(S)^2}. \tag{12}$$

4 Stochastic Radiance Estimate

In order to explain our new formulation, we first describe how to compute the average radiance value using the original progressive radiance estimate. Suppose that we have n sampled positions over the region S, $\vec{x}_1, \ldots, \vec{x}_n$. Using a Monte Carlo estimation, the original progressive radiance estimate can approximate the average radiance value in the region S as:

$$\begin{aligned} L(S, \vec{\omega}) &= \frac{1}{\|S\|} \int_S L(\vec{x}, \vec{\omega}) \, \mathrm{d}\, \mu(S) \\ &= \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} L(\vec{x}_k, \vec{\omega}) \approx \frac{1}{n} \sum_{k=1}^{n} L(\vec{x}_k, \vec{\omega}) \\ &= \frac{1}{n} \sum_{k=1}^{n} \lim_{i \to \infty} \frac{\tau_i(\vec{x}_k, \vec{\omega})}{N_e(i) \pi R_i(\vec{x}_k)^2}. \end{aligned} \tag{13}$$

We consider uniform sampling of $\vec{x}_k$ for brevity of discussion throughout the paper. Using non-uniform sampling follows the same discussion with premultiplied weight for $L(\vec{x}_k, \vec{\omega})$. This approach is not scalable for a large n because the progressive radiance estimate needs to keep track of statistics at each radiance sample (i.e., storing n sets of statistics in total). Moreover, the memory requirement for computing the correct average radiance value is unbounded because n needs to be infinite. The goal of stochastic progressive radiance estimate is to compute the correct average radiance value without storing infinite sets of photon statistics. The rest of this section describes how our formulation achieves this goal.

Our formulation assumes that the initial radius R_0 is constant within S, and the value of α is also constant within S. R_0 and α can still vary between different S (e.g., different R_0 per pixel). In addition, we only consider non-adaptive photon tracing (i.e., the photon tracing strategy is not affected by the photon statistics). Under these assumptions, we obtain the following equation:

$$\begin{aligned} R_{i+1}(\vec{x}) &= R_i(\vec{x})\sqrt{\frac{N_i(\vec{x}) + \alpha M_i(\vec{x})}{N_i(\vec{x}) + M_i(\vec{x})}} \\ &= R_i(\vec{x})\sqrt{\frac{C_N L(\vec{x}) R_i(\vec{x})^2 + \alpha C_M L(\vec{x}) R_i(\vec{x})^2}{C_N L(\vec{x}) R_i(\vec{x})^2 + C_M L(\vec{x}) R_i(\vec{x})^2}} \\ &= R_i(\vec{x}) C_P, \end{aligned} \tag{14}$$

where C_N, C_M, and C_P are constants independent of $\vec{x}$. This equation states that the rate of radius reduction is independent of the position $\vec{x}$ in S, thus $R_i(\vec{x})$ itself is also independent of $\vec{x}$ if R_0 is constant. We used the property that the number of new photons and local photon count $N_i(\vec{x})$ are both proportional to the search area $\pi R_i(\vec{x})^2$ and its true radiance $L(\vec{x})$ (or true photon density in general). In practice, this equation is only approximately true because $N_i(\vec{x})$ and $M_i(\vec{x})$ are stochastic variables. However, we found that it is reasonably true as we see that the reduction rate of the radius is almost constant in the corresponding graph of radius in the original PPM paper [Hachisuka et al. 2008].

4.1 Shared Radius

Based on the observation above, we can use a single radius value $R_i(\vec{x}_0)$ instead of $R_i(\vec{x}_k)$, in order to compute the average.

$$\frac{1}{n}\sum_{k=1}^{n} \lim_{i\to\infty} \frac{\tau_i(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(\vec{x}_k)^2} = \frac{1}{n}\sum_{k=1}^{n} \lim_{i\to\infty} \frac{\tau_i(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(\vec{x}_0)^2}. \tag{15}$$

This formulation removed the dependency of $R(\vec{x})$ on $\vec{x}$. This section describes how the dependency on an arbitrary location $\vec{x}_0$ can be further removed by using the shared radius $R_i(S)$ from Equation 9. Using the shared radius, the average radiance value is computed as:

$$\begin{aligned} L(S, \vec{\omega}) &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} L(\vec{x}_k, \vec{\omega}) \\ &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \lim_{i\to\infty} \frac{\tau_{R(S),i}(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(S)^2}. \end{aligned} \tag{16}$$

where $\tau_{R(S),i}(\vec{x}_k, \vec{\omega})$ is modified accumulated flux by changing the radius to $R_i(S)$. We show how this equation is derived in the following. Since accumulated flux is proportional to the area of the search region $\pi R_i(\vec{x}_0)^2$, $\tau_{R(S),i}(\vec{x}_k, \vec{\omega})$ is defined as:

$$\tau_{R(S),i}(x_k, \vec{\omega}) = \frac{R_i(S)^2}{R_i(x_0)^2}\tau_i(\vec{x}_k, \vec{\omega}). \tag{17}$$

Using this equation, we obtain:

$$\begin{aligned} \lim_{i\to\infty} \frac{\tau_{R(S),i}(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(S)^2} &= \lim_{i\to\infty} \frac{R_i(S)^2}{R_i(\vec{x}_0)^2} \frac{\tau_i(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(S)^2} \\ &= \lim_{i\to\infty} \frac{R_i(S)^2}{R_i(\vec{x}_0)^2} \frac{R_i(\vec{x}_0)^2}{R_i(S)^2} \frac{\tau_i(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(\vec{x}_0)^2} \\ &= C_R \frac{1}{C_R} L(\vec{x}_k, \vec{\omega}) = L(\vec{x}_k, \vec{\omega}). \end{aligned} \tag{18}$$

The last step is valid only if $\lim_{i\to\infty} R_i(S)^2 / R_i(\vec{x}_0)^2 = C_R$ with a non-zero constant C_R. In other words, $R_i(S)$ should be reduced on the same order of $R_i(\vec{x}_0)$ in order this step to be valid. We provide the derivations that show this is true in Appendix A. Note that an arbitrary $R_i(S)$ does not necessarily satisfy this condition, and our choice of $R_i(S)$ is crucial in this step. As a result of this derivation, we can replace each radius $R_i(\vec{x}_k)$ by a single shared radius $R_i(S)$, and its estimated radiance value still converges to the correct radiance value at $\vec{x}_k$. The shape of search region is still a sphere but its radius is defined by the shared radius $R_i(S)$.

4.2 Shared Accumulated Flux

Equation 16 still requires storing $\tau_i(\vec{x}_k, \vec{\omega})$ at each $\vec{x}_k$ in order to compute the correct average radiance value over S. This approach needs to keep track of an infinite number of accumulated flux values and positions over S, which is infeasible. Approximation using a fixed number of $\tau_i(\vec{x}_k, \vec{\omega})$ is possible, but this approach is not consistent. The shared accumulated flux value in Equation 11 solves this problem by storing a single accumulated flux value with a random position $\vec{x}_i$ at each photon pass. This section describes how this can be achieved. Replacing $\tau_i(\vec{x}_k, \vec{\omega})$ by the shared accumulated flux $\tau_i(S, \vec{\omega})$, we obtain the following estimate:

$$L(S, \vec{\omega})' = \lim_{i\to\infty} \frac{\tau_i(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2}. \tag{19}$$

In order to show $L(S, \vec{\omega})' = L(S, \vec{\omega})$, we take the difference as:

$$\begin{aligned} & L(S, \vec{\omega}) - L(S, \vec{\omega})' \\ &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \left(\lim_{i\to\infty} \frac{\tau_{R(S),i}(\vec{x}_k, \vec{\omega})}{N_e(i)\pi R_i(S)^2} \right) - \frac{\tau_{R(S),i}(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2} \\ &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \lim_{i\to\infty} \frac{\tau_{R(S),i}(\vec{x}_k, \vec{\omega}) - \tau_i(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2} \\ &= \lim_{n\to\infty} \lim_{i\to\infty} \frac{E_i}{N_e(i)\pi R_i(S)^2}, \end{aligned} \tag{20}$$

where we defined $E_i = 1/n \sum_{k=1}^{n} (\tau_{R(S),i}(\vec{x}_k, \vec{\omega}) - \tau_i(S, \vec{\omega}))$. This difference converges to zero if the denominator $N_e(i)\pi R_i(S)^2$ diverges with an infinite number of passes and $|E_i|$ is bounded. The former is true because $\lim_{i\to\infty} N_e(i)\pi R_i(\vec{x}_0)^2$ is divergent (one of the conditions of consistency) and $R_i(S)^2 / R_i(\vec{x}_0)^2$ is a non-zero constant as in Section 4.1., which shows that $\lim_{i\to\infty} N_e(i)\pi R_i(S)^2$ is also divergent. We provide the details how $|E_i|$ is bounded in Appendix B. Although the final result looks simple, our choice of $\tau_i(S, \vec{\omega})$ is again crucial to bound $|E_i|$. Finally, we can estimate the correct average radiance value as:

$$L(S, \vec{\omega}) = L(S, \vec{\omega})' = \lim_{i\to\infty} \frac{\tau_i(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2}. \tag{21}$$

	Triangles	Rendering time [min]	PPM passes	SPPM passes
Cornell Box (Figure 4)	38	50	899	742
Cornell Box with Wall lights (Figure 4)	7660	50	234	220
Furry Bunny (Figure 6)	371247	132	1722	1197
Transparent Dices (Figure 6)	370860	110	287	195
Alarm clocks (Figure 7)	119856	480	1101	1202
Tools (Figure 1)	56486	200	353	484

Table 1: *Rendering statistics of our experiments. We show a single rendering time for each scene because both PPM and SPPM used the same rendering time. PPM passes are the numbers of photon tracing passes, and SPPM passes are the numbers of photon tracing passes as well as the numbers of distributed ray tracing passes. We used 500,000 emitted photons per pass in both methods. SPPM usually performs a less number of passes in the same rendering time because of the additional cost of the distributed ray tracing pass. However, SPPM can perform more passes in the alarm clocks scene and the tools scene, where PPM needed to use multiple hit points per pixel (16 hit points per pixel) to achieve distributed ray tracing effects.*

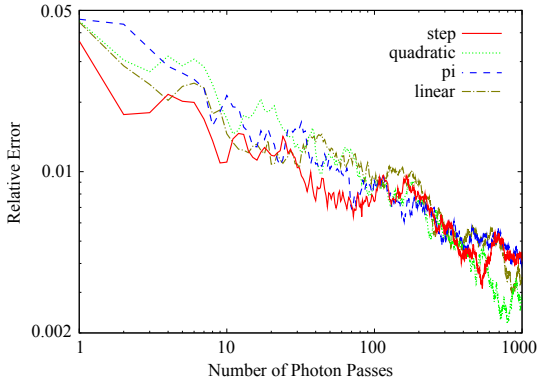
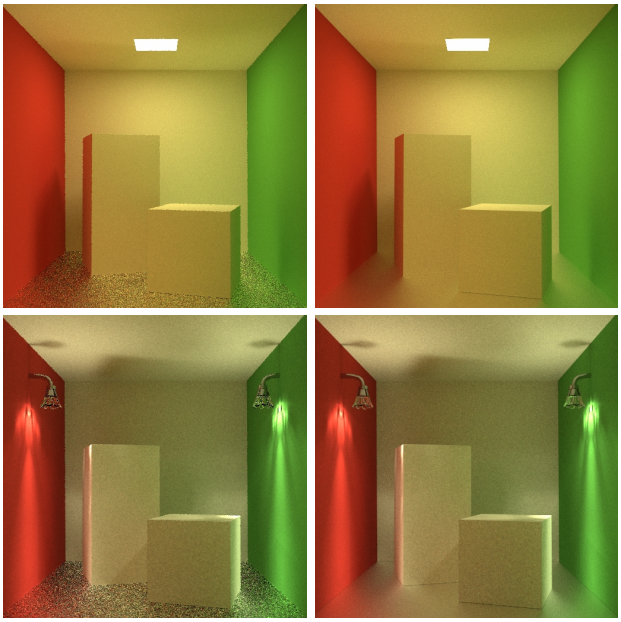


Figure 3: *Error plot of 1D integration tests. We performed numerical integration of functions using our method. The functions are* $y0(x) = 0.75 + 0.25\,\mathrm{sgn}(x - 0.5)$ *(step),* $y1(x) = x^2$ *(quadratic),* $y2(x) = 4\sqrt{1 - x^2}$ *(pi), and* $y3(x) = x$ *(linear), where the range of integration is* $x \in [0, 1]$ *for all functions. We compute the relative error as* $E(y) = \left|\frac{Y - y}{Y}\right|$*, where* Y *is the analytical value of integration and* y *is a current estimate. The plot is using the average over 10 different random number sequences. The errors converge to zero as the number of photon passes increases.*



PPM SPPM

Figure 4: *Cornell box with a glossy floor. The BRDF of the floor is the modified Phong model with a glossiness value of* 25*. The results of PPM are noisy because glossy reflections cause large variations of photon contributions to viewing directions. The bottom row replaced the area light source in the original Cornell box with two wall lights. Even though this change is simple from a user's perspective, it makes unbiased methods inefficient because of highly glossy reflections of caustics from the wall lights. SPPM is still as robust as PPM for this illumination setting, while rendering glossy reflections with less noise in the same rendering time.*

5 Results

In this section, we show results based on our implementation of progressive photon mapping (PPM) and our stochastic progressive photon mapping (SPPM). The implementation of SPPM is almost the same as PPM. One change is generating a set of new hit points by distributed ray tracing [Cook et al. 1984] after each photon pass. Another change is to assign shared statistics to each pixel, not each hit point. This is because the statistics are shared over the region S, and S is usually assigned to each pixel (e.g., a pixel footprint).

All of our test scenes have been rendered on a 2.4GHz Intel Core 2 Q6600 using one core. The resolution of the images is 640×480, except for the bunny scene and the Cornell box scenes which use 512×512. All rendering comparisons are equal time, except for the progressive sequences in Figure 5. In all the scenes, each photon pass traced 500,000 photons and α is set to 0.7. Table 1 summarizes the statistics of our experiments.

Figure 3 shows a numerical validation of SPPM with a 1D function integration. We used 1D functions where the results of integrations over $[0, 1]$ are known, and performed numerical integrations by estimating the average values over $S = [0, 1]$ with SPPM. In each photon pass, photons (samples) are generated using rejection sampling so that they are distributed according to the function. We used 50,000 photons per pass and took the average error over 10 different runs in this experiment. As we can see, errors of SPPM are converging to zero as the number of photon tracing passes increases.

In order to highlight the improvement over PPM, we rendered the Cornell box with a glossy floor as in Figure 4. The floor uses the modified Phong mode with a glossiness value of 25. Note that the results with PPM have significant noise on the floor. PPM is not efficient for glossy reflections because incoming directions of photons that contribute to the viewer direction are narrow. SPPM efficiency handles glossy reflections by tracing once bounce camera rays according to the BRDF and computing the hemispherical integration, similar to the final gathering step in standard photon mapping [Jensen 1996].

Figure 5 shows a progressive sequences of renderings for the Cornell box scene and the RMS errors compared to a reference result. For the reference result, we used the converged result generated using PPM with a large number of photon passes. We show images from both PPM and SPPM using the same number of photon passes, not equal time comparisons. Note that each pass of SPPM takes ap-

proximately 10 percent longer rendering time than PPM, because of additional distributed ray tracing per pass. However, SPPM can render visually pleasing results with a smaller number of photon passes in comparison to PPM, which makes SPPM worth the additional computational cost per pass. In addition, the RMS errors of SPPM are consistently lower than PPM as shown in the graph.

Figure 6 shows the applications to anti-aliasing and motion blur. PPM used 1 sample per pixel to have equal memory consumption with SPPM. In order to correctly render motion blur with PPM, we sampled the time when hit points are generated (16 samples per pixel are used). The sampled time value is assigned to each hit point so that each photon contributes to hit points in the same time sample. Using the same rendering time, SPPM can include anti-aliasing and motion blur with a constant amount of memory consumption independent of the number of samples per pixel. PPM needs an increasing amount of memory in order to increase the number of samples per pixel.

Figure 7 shows a rendering with depth-of-field. PPM used 16 hit points (i.e., 16 radiance samples) per pixel, which consumed approximately 1GB of memory in our implementation. Note that the scene is dominated by SDS paths because all the illumination is due to the desk lamp with a light bulb outside the view, and we observe the scene through a lens to achieve depth-of-field. Although PPM can handle SDS paths, the result is noisy because the number of radiance samples per pixel is insufficient to remove noise due to depth-of-field. SPPM renders the same scene with less noise using 16 times less memory consumption.

We also provide the comparison with bidirectional path tracing (BDPT) with multiple importance sampling [Veach and Guibas 1995] in this scene. The comparison confirms that this scene cannot be rendered efficiently by BDPT due to SDS paths. Since the robustness of SPPM to SDS paths is the same as PPM, for more comparisons with unbiased methods, readers might want to refer to the PPM paper [Hachisuka et al. 2008].

Finally, we show a rendering of a scene with depth-of-field and glossy reflections in Figure 1. PPM used 16 hit points per pixel as before. The bolts and plier used the modified Phong model with the glossiness value of 100. The entire scene is illuminated by the flashlight in front, which causes highly glossy reflections of caustics on the metallic parts of plier and bolts. The result with PPM suffers from both noise due to glossy reflections and noise due to depth-of-field. Again, note that PPM consumed approximately 16 times more memory than SPPM because PPM uses 16 samples per pixel. In the same rendering time with smaller memory consumption, SPPM renders this scene with less noise.

6 Conclusion

We have presented a new formulation of progressive photon mapping, called stochastic progressive photon mapping, that makes it possible to compute the correct average radiance value over a region. We modify progressive photon mapping by adding a new distributed ray tracing pass that generates new hit points after each photon pass. The photon statistics of the new hit points is taken directly from the previous hit point for each pixel, and this is the key insight that makes it possible to compute the correct average radiance for a region rather than at a point. Our results show that stochastic progressive photon mapping is robust in scenes with complex illumination settings including distributed ray tracing effects, such as depth-of-field, motion blur, and glossy reflections/refractions.

We believe that applications of stochastic progressive radiance estimation outside rendering are possible because our formulation can be thought as a general density estimation framework that can compute the correct average density value. One limitation is that the current formulation is restricted to the average radiance on a surface as the method is based on PPM. Extending the method to the average radiance value from volume would be useful because it is required for rendering participating media. Although we have shown that the additional errors of our formulation over progressive photon mapping are theoretically bounded and converging, establishing more accurate error estimates would be interesting for future work.

Acknowledgments

We would like to thank Youichi Kimura (Studio Azurite) for providing the flashlight model, Will Chang, Wan-Yen Lo, Marios Papas, and Iman Sadeghi (UCSD Graphics Lab) for discussions and comments on the draft. We would also like to thank Antoine Bouthors (Weta Digital), Wojciech Jarosz (Disney Research, Zürich), and Leonhard Grünschloß (Nvidia) for further comments on the draft. The clock, bolts, and plier models are from ShareCG.com. This work was supported by ATI fellowship 2008.

References

CAMMARANO, M., AND JENSEN, H. W. 2002. Time dependent photon mapping. In *Rendering Techniques*, Eurographics Association, S. Gibson and P. E. Debevec, Eds., vol. 28 of *ACM International Conference Proceeding Series*, 135–144.

CLINE, D., TALBOT, J., AND EGBERT, P. 2005. Energy redistribution path tracing. *ACM Trans. Graph. (SIGGRAPH Proceedings) 24*, 3, 1186–1195.

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In *Computer Graphics (SIGGRAPH Proceedings)*, vol. 3(18), 137–45.

DUTRÉ, P., BEKAERT, P., AND BALA, K. 2006. *Advanced Global Illumination (2nd edition)*. A K Peters.

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

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. The beam radiance estimate for volumetric photon mapping. *Comput. Graph. Forum 27*, 2, 557–566.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, Springer-Verlag, London, UK, 21–30.

KAJIYA, J. T. 1986. The rendering equation. *Computer Graphics (SIGGRAPH Proceedings) 20*, 4, 143–150.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, H. P. Santo, Ed., 145–153.

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

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

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Computer Graphics (SIGGRAPH Proceedings)*, 65–76.

WASSERMAN, L. 2006. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

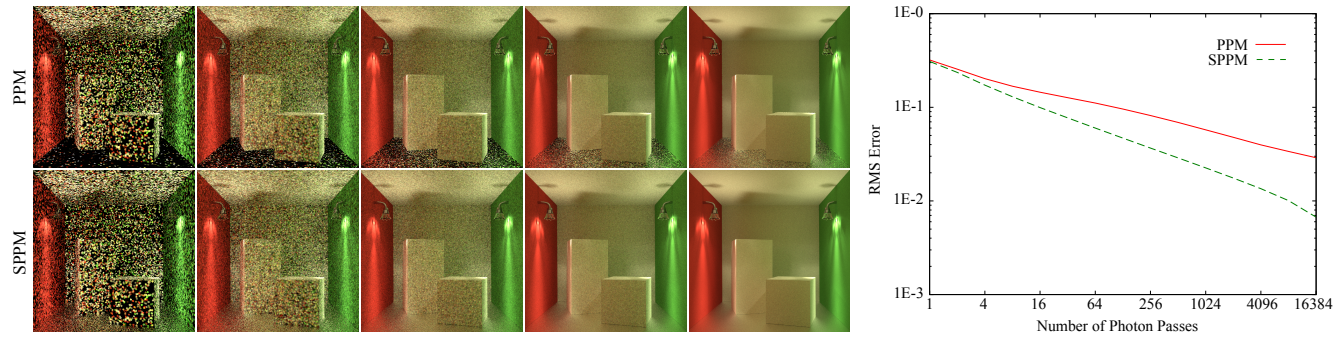


Figure 5: *Progressive sequences of rendering of Cornell box and the RMS errors. The number of photon passes of the images is* 1, 8, 64, 512, *and* 4096 *correspondingly. The graph shows the RMS errors from the converged result with PPM. The result with SPPM converges visually pleasing results with a smaller number of photon passes, and the RMS errors are consistently lower than the result with PPM. The rendering time of SPPM is approximately* 10 *percent longer than that of PPM for the same number of photon passes.*

Figure 6: *Furry bunny illuminated by the skylight and moving transparent dices with motion blur. The dices are illuminated by the sunlight, and blur of caustics and shadows is due to motion blur of the dices. The close-ups compare the results with PPM (left columns) and SPPM (right columns), and we show the entire rendered images with SPPM. The comparisons are equal time and equal memory consumption (i.e., PPM used* 1 *sample per pixel).*

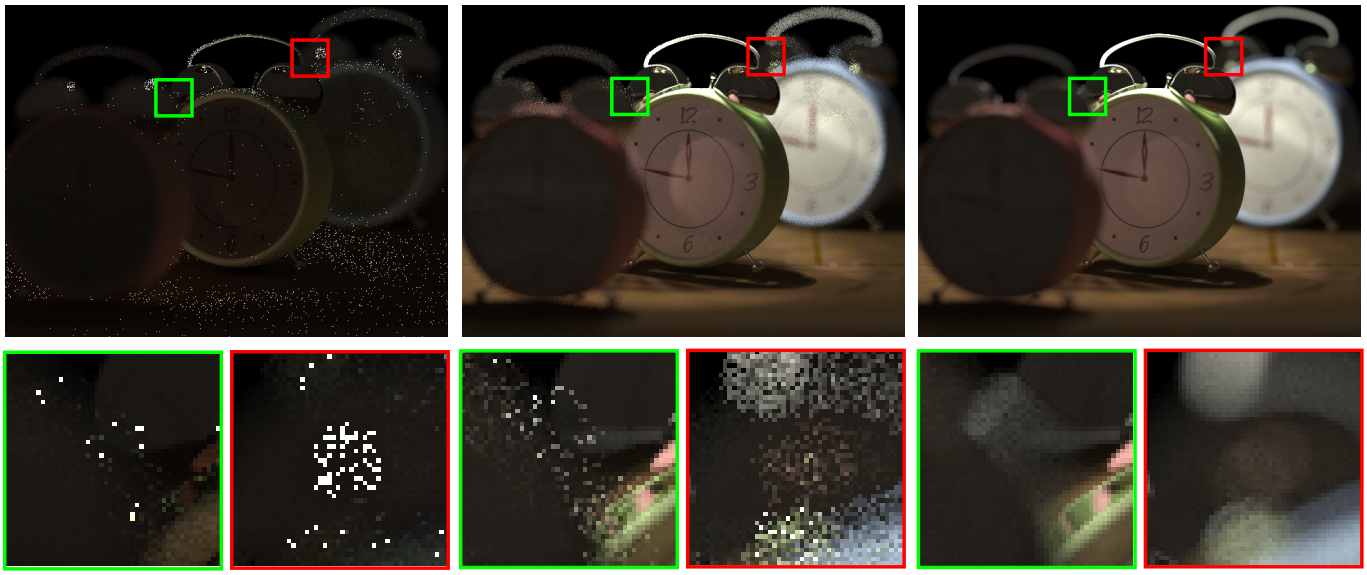


Figure 7: *Alarm clocks illuminated by a desk lamp. The desk lamp with a light bulb is outside the view and illuminates the clocks. The scene is rendered using a thin lens model for depth-of-field. The images are rendered with BDPT (left), PPM (middle), and SPPM (right) using the same rendering time. The combination of the lens for depth-of-field and caustics from the desk light makes the entire scene dominated by SDS paths, which is difficult to render with unbiased methods such as BDPT. BDPT used* 11406 *paths per pixel, but the image is dark and contains noisy bright pixels. PPM can handle such an illumination setting, but the close-ups show rendering of depth-of-field causes visually noticeable noise due to the fixed number of samples per pixels. SPPM robustly handles illumination by the desk lamp as well as depth-of-field in the same rendering time with less memory consumption.*

A Convergence of Ratio of Radius

We show that the ratio $\lim_{i\to\infty} R_i(S)^2/R_i(\vec{x}_0)^2$ is convergent and non-zero in the following. We first expand the ratio using its definition:

$$\lim_{i\to\infty}\frac{R_i(S)^2}{R_i(\vec{x}_0)^2}=\prod_{j=0}^{\infty}\frac{(N_j(S)+\alpha M_j(\vec{x}_j))(N_j(\vec{x}_0)+M_j(\vec{x}_0))}{(N_j(S)+M_j(\vec{x}_j))(N_j(\vec{x}_0)+\alpha M_j(\vec{x}_0))} \tag{22}$$

In the following derivations, we use the notations $N_S = N_j(S)$, $N_0 = N_j(\vec{x}_0)$, $M_j = M_j(\vec{x}_j)$, and $M_0 = M_j(\vec{x}_0)$ for readability. Using these notations, we further expand the equation and obtain the upper bound as:

$$\begin{aligned}
&=\prod_{j=0}^{\infty}\frac{(N_S+\alpha M_j)(N_0+M_0)}{(N_S+M_j)(N_0+\alpha M_0)}\\
&=\prod_{j=0}^{\infty}\left(1+\frac{(1-\alpha)(N_SM_0-N_0M_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}\right)\\
&=\prod_{j=0}^{\infty}\left(1+\frac{(1-\alpha)N_0M_0(\frac{N_S}{N_0}-P_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}\right)\\
&=\prod_{j=0}^{\infty}(1+Q_j)\,.
\end{aligned} \tag{23}$$

where $P_j = \frac{M_j}{M_0}$ and we defined Q_j as:

$$Q_j=\frac{(1-\alpha)N_0M_0(\frac{N_S}{N_0}-P_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}. \tag{24}$$

Taking the logarithm of both sides, this infinite product converges if and only if the infinite sum

$$Q=\sum_{j=0}^{\infty}Q_j \tag{25}$$

converges as each term of the infinte product is always positive. Note that Q_j is a random variable. Assuming non-adaptive photon tracing, we can consider $M_0 = M_j(\vec{x}_0) = 1$ for a large enough j. We thus obtain

$$\lim_{j\to\infty}\frac{N_S}{N_0}=\lim_{j\to\infty}\frac{\alpha\sum_j M_j}{\alpha\sum_j M_0}=\lim_{j\to\infty}\frac{\sum_j M_0P_j}{\sum_j M_0}=\mathrm{E}[P_j], \tag{26}$$

so the numerator $\frac{N_S}{N_0} - P_j$ is symmetric around zero, therefore, $\mathrm{E}[Q_j] = 0$. Furthermore, we obtain the following:

$$\sum_{j=0}^{\infty}\mathrm{Var}[Q_j^2]=\sum_{j=0}^{\infty}\mathrm{E}[Q_j^2]<\sum_{j=0}^{\infty}\frac{C_1}{(j+1)^2}<\infty, \tag{27}$$

where C_1 is a constant because the numerator of Q_j is bounded. This derivation also uses the fact that the lower bound of N_S is $\alpha(j+1)$ (i.e., only one photon is captured), and the upper bound of M_0 is $Ne(1)$ (i.e., all the emitted photons are captured).

Using $\mathrm{E}[Q_j] = 0$ and $\sum_{j=0}^{\infty}\mathrm{Var}[Q_j^2] < \infty$ with Kolmogorov's one series theorem, the infinite sum Q almost surely converges. Therefore, the infinite product also almost surely converges. We then show that the ratio is non-zero by showing the reciprocal of the ratio is bounded. Namely, we show that $\lim_{i\to\infty} R_i(\vec{x}_0)^2/R_i(S)^2$ is convergent. Similar to before, we expand the ratio as follows:

$$\begin{aligned}
\lim_{i\to\infty}\frac{R_i(\vec{x}_0)^2}{R_i(S)^2}&=\prod_{j=0}^{\infty}\frac{(N_S+M_j)(N_0+\alpha M_0)}{(N_S+\alpha M_j)(N_0+M_0)}\\
&=\prod_{j=0}^{\infty}\left(1+\frac{(1-\alpha)(N_0M_j-N_SM_0)}{N_SN_0+N_SM_0+\alpha M_jN_0+\alpha M_jM_0}\right)\\
&=\prod_{j=0}^{\infty}(1+Q'_j)\,.
\end{aligned} \tag{28}$$

The rest of the derivation is the same as before, but with the opposite sign of random variables and use Q'_j instead of Q_j.

B Bound of $|E_i|$

We show that $|E_i|$ is bounded by a constant. In the following derivation, we first consider $n|E_i|$ for the purpose of discussion. We expand $n|E_i|$ using the definition of $\tau_i(S,\vec{\omega})$ and $\tau_{R(S),i}(\vec{x}_k,\vec{\omega})$:

$$\begin{aligned}
n|E_i|&=\left|\sum_{k=1}^{n}\left(\tau_{R(S),i}(\vec{x}_k,\vec{\omega})-\tau_i(S,\vec{\omega})\right)\right|\\
&=\left|\sum_{k=1}^{n}\left(\sum_{j=0}^{i}\Phi_j(\vec{x}_k)-\sum_{j=0}^{i}\Phi_j(\vec{x}_j)\right)\right|.
\end{aligned} \tag{29}$$

Note that we can use the same $\Phi_j(\vec{x})$ to expand both $\tau_{R(S),i}(\vec{x}_k,\vec{\omega})$ and $\tau_i(S,\vec{\omega})$ because both accumulated flux values use the same search radius $R(S)$ as a result of Section 4.1. Since photon flux $\Phi_j(\vec{x})$ is proportional to the radiance value $L(\vec{x})$ (i.e., $\Phi_j(\vec{x})$ divided by the area is radiance) and radius is the same everywhere in S (Equation 14), we can further expand the above equation into:

$$\begin{aligned}
&=\frac{1}{L(\vec{x}_0)}\left|\sum_{k=1}^{n}\left(\sum_{j=0}^{i}L(\vec{x}_k)\Phi_j(x_0)-\sum_{j=0}^{i}L(\vec{x}_j)\Phi_j(x_0)\right)\right|\\
&=\frac{1}{L(\vec{x}_0)}\left|\sum_{k=1}^{n}\sum_{j=0}^{i}\Phi_j(\vec{x}_0)(L(\vec{x}_k)-L(\vec{x}_j))\right|\\
&=\frac{1}{L(\vec{x}_0)}\left|\sum_{j=0}^{i}\Phi_j(\vec{x}_0)\sum_{k=1}^{n}(L(\vec{x}_k)-L(\vec{x}_j))\right|.
\end{aligned} \tag{30}$$

$\Phi_j(\vec{x}_k) = \frac{L(\vec{x}_k)\Phi_j(x_0)}{L(\vec{x}_0)}$ is only approximately true because $\Phi_j(\vec{x}_k)$ is estimated using a finite number of photons per pass. However, this error does not diverge as $j \to \infty$ (i.e., we at least have one photon per pass), so we can ignore this error in this derivation. Since photon flux $\Phi_j(\vec{x})$ is monotonically decreasing as $j \to \infty$ and $i = n$ in our formulation, we obtain the upper bound as:

$$\begin{aligned}
n|E_i|&\le\frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\sum_{j=0}^{i}\left|\sum_{k=1}^{n}(L(\vec{x}_k)-L(\vec{x}_j))\right|\\
&=\frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\sum_{j=0}^{n}\left|\sum_{k=1}^{n}L(\vec{x}_k)-nL(\vec{x}_j)\right|
\end{aligned} \tag{31}$$

Assuming the variation of $L(\vec{x})$ is bounded (i.e., $0 \le L(\vec{x}) < \infty$), the double summation above can be written as $C_2 n$, where C_2 is a constant. Dividing the both side by n, we obtain:

$$|E_i|\le\frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\frac{1}{n}C_2n=\frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}C_2, \tag{32}$$

Therefore, $|E_i|$ is bounded by a constant.