Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Diffuse Global Illumination in Particle Spaces

Byungjoon Chang · Sanghun Park · Insung Ihm

Received: date / Accepted: date

Abstract Despite substantial efforts in recent years to accelerate rendering methods, the traditional method, based on a combination of recursive ray tracing (RT), photon mapping (PM), and final gathering (FG), is still regarded as computationally intensive. In this paper, we propose a practical ray tracing model that can be readily implemented on a graphics processing unit (GPU) to provide high-speed generation of global illumination, whose quality is comparable to that generated through the traditional time-consuming RT/PM/FG rendering method. Our method employs two particle spaces to generate computationally intensive diffuse interreflection more efficiently. The complexity of light transport within a scene is simulated in one particle space by using indirect light scattering and gathering operations. The calculation that estimates the reflected radiance caused by diffuse interreflection is optimized by using a second particle space, where only the radiance required for final rendering can be rapidly approximated, based on the simulated light flux in the first particle space. We present several example scenes to demonstrate that our ray tracing scheme enables the use of a rendering pipeline that fully exploits the computing architecture of current manycore processors to reproduce effective high-quality global illumination.

Keywords global illumination · diffuse interreflection · particle space · ray tracing · photon mapping and final gathering · multiple bounces

B. Chang
Graphics Laboratory, Digital Media and Communications R&D Center, Samsung Electronics, Suwon 443-742, Republic of Korea
Tel.: +82-31-279-8158, Fax: +82-31-279-1295
E-mail: bj81.chang@samsung.com

S. Park
Department of Multimedia, Dongguk University, Seoul 100-715, Republic of Korea
Tel.: +82-2-2260-3765, Fax: +82-2-2260-3766
E-mail: mshpark@dongguk.edu

I. Ihm
Department of Computer Science and Engineering, Sogang University, Seoul 121-742, Republic of Korea
Tel.: +82-2-705-8493, Fax: +82-2-704-8273
E-mail: ihm@sogang.ac.kr

1 Introduction

1.1 Background

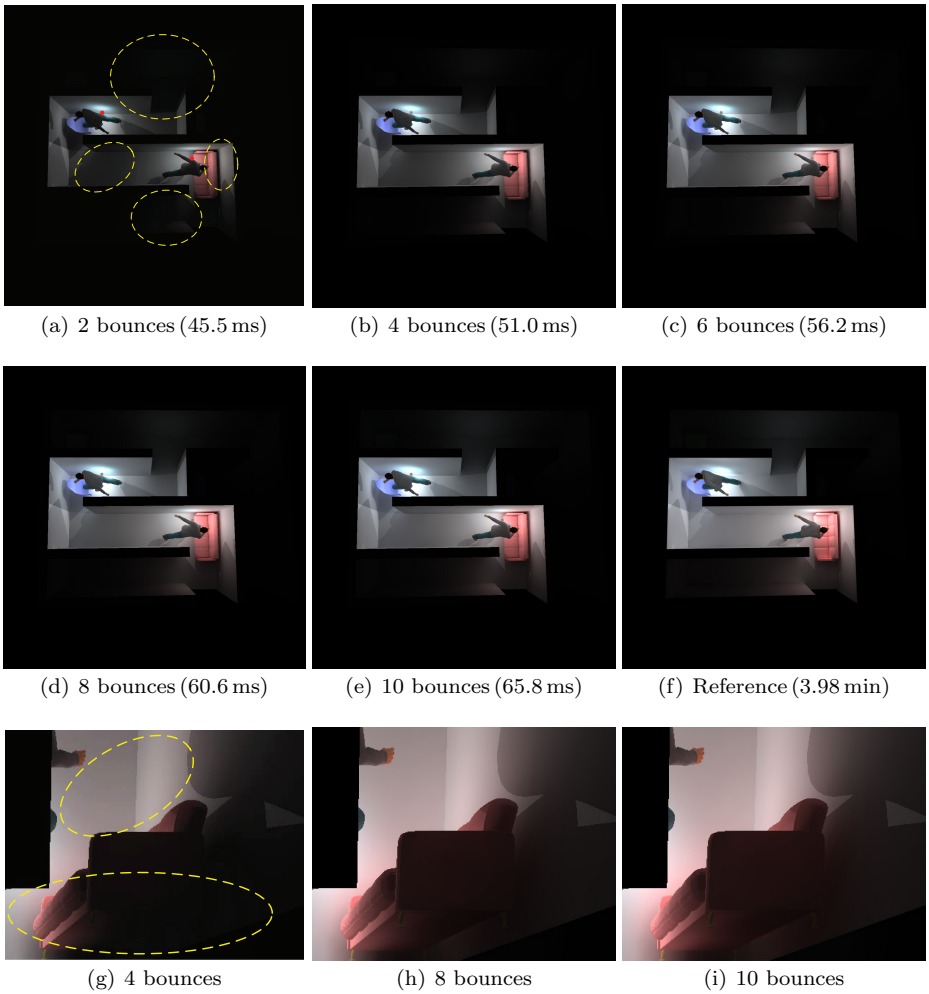
The faithful reproduction of light transport within a scene is an essential element of physically based photorealistic rendering. One of the most important, but computationally intensive, aspects of light transport rendering is diffuse interreflection, which refers to the reflection of incident light arriving from diffuse surfaces after one or more bounces from other diffuse surfaces. Since the path tracing method was introduced to the computer graphics community by Kajiya [15], a number of effective global illumination algorithms, such as radiosity and photon mapping, have been developed over the years to generate indirect illumination effects. Recent advances in massively parallel manycore processing technology have further intensified the demand for interactive, or real-time, reproduction of global illumination in complex scenes. An essential requirement for achieving this elusive goal is the development of an effective computational model for light transport that can run efficiently on the computing architecture of current manycore processors without significantly sacrificing the quality of the global illumination produced.

A challenging aspect of diffuse interreflection is the inherent light scattering property of diffuse surfaces, which makes it difficult to follow transported light precisely. To produce an acceptable outcome, it is often necessary to trace a huge number of light paths of similar importance, without being able to focus on a smaller subset. In addition, as in the difficult situations exemplified by Figure 1, it often happens that a large number of indirect light bounces must be processed to produce adequate indirect illumination effects. Many of the recent interactive global illumination algorithms rely on heuristic methods, such as hierarchical representations of indirect light or irradiance/radiance caching, and often limit themselves to a few light bounces for efficiency in computation (see the survey paper [26] for a summary of interactive global illumination techniques). In summary, despite substantial recent research efforts being directed toward the development of practical algorithms, producing high-quality diffuse interreflection interactively for complex scenes remains a challenge.

1.2 Our contribution

In this paper, we present an extended ray tracing model that aims to rapidly produce global illumination effects, whose quality is comparable to that generated using the computationally intensive final gathering technique. To develop a light transport model that fully harnesses the massively parallel computing power of current graphics processing units (GPUs), we discretize the light transport space into a particle-based discrete system, where indirect illumination is simulated using a simple three-stage GPU-friendly rendering algorithm (refer to Figure 4 for an overview of our method).

In the *geometry stage*, the input scene is first discretized into a collection of particles, called *area particles*, each of which represents a small area around its center (see Figure 2(a)). Then, via directional stochastic sampling over the hemisphere around each area particle, a small number of light paths, called *area particle links*, are constructed in the area particle space, which allows efficient *indirect light*



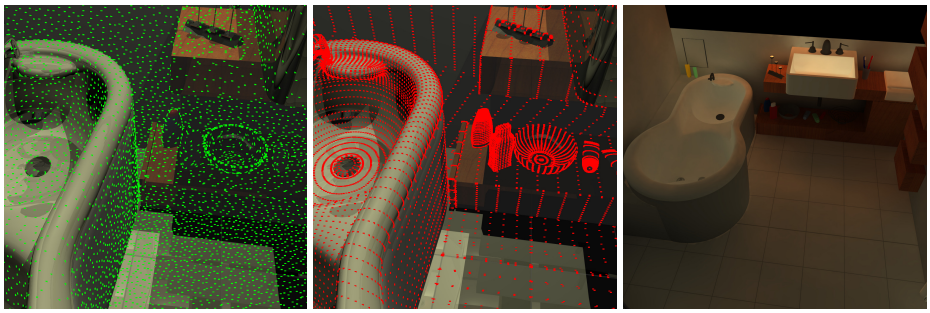
(a) 2 bounces (45.5 ms) (b) 4 bounces (51.0 ms) (c) 6 bounces (56.2 ms)

(d) 8 bounces (60.6 ms) (e) 10 bounces (65.8 ms) (f) Reference (3.98 min)

(g) 4 bounces (h) 8 bounces (i) 10 bounces

Fig. 1 Effect of multiple bounces of diffuse interreflection. In a complex situation such as this maze scene, which comprises 241,080 triangles and 2 lights, it is important to handle diffuse interreflection carefully to show subtle changes in indirect illumination. The timings (in milliseconds) are for an NVidia GeForce GTX 680 GPU to generate a $1,024 \times 1,024$ pixel image by full ray tracing after 6.43 seconds of the geometric stage of a computation that allowed free changes of viewpoint and lights. The reference image was generated by our GPU ray tracer implementing photon mapping with final gathering.

scattering in the discrete system. In addition, to minimize the overhead caused by repeated estimation of the reflected radiances at shading points in our photon-mapping-style algorithm, we identify another kind of particle, called a *radiance particle*. These particles are implicitly defined to exist at the vertices of the sub-triangles obtained by tessellating each triangle in the geometric models for the scene (Figure 2(b)). Each radiance particle is associated with a list of area particles, called a *radiance estimation list*, that includes the area particles whose fluxes are used in the radiance estimation. In the next *lighting stage* of the *indirect light*



(a) Area particles (b) Radiance particles (c) Diffuse interreflection

(d) Ours 1 (146.8 ms) (e) Ours 2 (293.0 ms) (f) Reference (9.76 min)

Fig. 2 Light transport in particle spaces. (a) The BATHROOM scene with 256,907 triangles discretized into 200,000 area particles for the efficient simulation of light transport on a GPU. (b) The reflected radiance from diffuse interreflection is based on the transported light flux in the area-particle space and estimated only for necessary radiance particles. (c) The estimated radiance at the radiance particles is then ray-traced to create diffuse interreflection on the surfaces of objects effectively. (d) & (e) The method is easily combined with a Whitted-style ray tracer with (d) single-pixel sampling and (e) adaptive supersampling [14] that allows antialiased, high-quality rendering. (f) Our results show only slight visual differences compared with the reference image, generated by our GPU ray tracer supporting photon mapping with final gathering.

scattering, the radiant energy from light sources is efficiently propagated among the area particles using the area particle links. In the final *ray tracing stage*, the small set of radiance particles actually involved in shading are first extracted via preliminary ray tracing. An additional light transport pass, called *indirect light gathering*, is then conducted, only for the area particles in the radiance estimation lists of the extracted radiance particles, to estimate the reflected radiance. This enables the production of high-quality diffuse interreflection that compares favorably to that achieved via the final gathering method (see Figure 2(c) for the diffuse interreflection generated using our method). Finally, a recursive ray tracer is run, whereby the diffuse interreflection at each ray-surface hit is calculated quickly by simple linear interpolation using the radiance of three surrounding radiance particles. See Figure 2 for a comparison of our results ((d) and (e)) with a reference image ((f)) generated by a GPU ray tracer that implements photon mapping with final gathering.

The proposed method adopts a *discrete light transport model* that is suited to straightforward and efficient implementation on a GPU. By using two particle spaces, the light scattering and light gathering computations, which correspond to the photon tracing and final gathering stages, respectively, are effectively combined to produce an interactive global illumination algorithm. The area particle links enable the efficient generation of the diffuse interreflection, which may often require many indirect light bounces. The refined area particle model also facilitates the removal of problematic visual artifacts such as the boundary bias. In addition, the radiance particles enable the effective production of low-cost high-quality global illumination on the fly during ray tracing.

Note that the idea of particle-based (or point-based) global illumination is not new. For example, Christensen [2] discretized the scene geometry using a point cloud of surface elements (surfels) with direct illumination values. The surfels were organized into an octree on the fly during rendering, where, for each octree node, a spherical harmonic representation of the radiant power emitted in the node was computed. The diffuse indirect illumination was then approximated by traversing the octree and rasterizing the color contribution from the visited nodes. Although very effective, this method is not well suited to rendering scenes that need to simulate many bounces of light propagation because the entire process must be iterated for every light bounce to update the radiosity of the surfels in the point cloud.

Lehtinen et al. [20] also simulated indirect illumination on point samples scattered across a scene. However, their work focused more on efficient computation of direct-to-indirect light transport for complex scenes based on a hierarchical function, and required up to 30 minutes of precomputation on a CPU to allow for changes in the dynamic viewpoint and lights. In contrast, our method aims to provide a global illumination scheme, which is much simpler in computational structure and is therefore better suited to GPUs. As a result, our method can generate the indirect illumination from multiple bounces by simply following the area particle links in parallel on the GPU. Furthermore, it usually takes less than 10 seconds to create a full set of area particles, area particle links, and radiance estimation lists from an input scene comprising a few hundreds of thousands of triangles, which then permit an interactive response to changes in the dynamic viewpoint and lights.

Furthermore, like our method, Schmitz et al. represented discretely-sampled explicit visibility using a similar link structure to accelerate radiance propagation through the scene by carrying out repeated gathering operations at the vertices of geometric models [30]. However, our light transport algorithm differs in that it employs both scattering and gathering operations via geometry-decoupled area particles for optimizing the higher-bounce light propagation process. The effectiveness of performing both scattering and gathering operations on the fly to compute diffuse global illumination is discussed with examples in Section 5.1.

2 Related work

There exists a vast body of research on computing global illumination, so we only consider the most relevant work in this section. Jensen developed a two-pass algorithm, known as photon mapping, as a practical method for solving the rendering

equation [15], which effectively generates the full range of global illumination effects [12]. Final gathering stochastically samples the incident radiance at each shading point; this method is easily combined with photon mapping to eliminate any visually objectionable blotchiness that might result from directly visualizing insufficiently sampled photons [38]. The combination of recursive ray tracing, photon mapping, and final gathering routinely generates high-quality global illumination, but this method is usually computationally intensive, because of the large amount of visibility computation required for photon tracing and radiance gathering, the repetitive photon queries required for radiance estimation, and the use of large numbers of photons and gathering rays during high-quality rendering.

Several caching and interpolation techniques have been proposed to reduce the cost of performing final gathering for only a sparse set of surface points [38,37,32, 17,1]. These methods are very effective on the CPU, but they are not well suited for efficient GPU implementation because of their inherent computing structures, which usually demand sequential queries and cache updates. In contrast, the first GPU implementation of photon mapping by Purcell et al. [25] and its variants greatly accelerated photon mapping and final gathering on the GPU (refer to [19, 9,33] for some examples of early work). Zhou et al. developed a fast GPU-assisted kd-tree algorithm for fast ray tracing and photon mapping [40]. Wang et al. extended this algorithm by adding several complex global illumination effects at interactive speed, which reduced the rendering cost by performing the final gathering operation only at cluster centers [36]. Fabianowski and Dingliana extended the idea of photon differentials to interactively handle diffuse reflections for dynamic lights and cameras [8]. Ritschel et al. developed a micro-rendering technique to accelerate final gathering on the GPU, which allowed efficient gathering of radiance using tiny micro-buffers [27]. Image space photon mapping methods were also presented by McGuire and Luebke [21] and Yao et al. [39] to achieve interactive global illumination in the context of rasterization-based rendering pipelines.

Instant radiosity, presented by Keller [16], and its successors comprise an important class of interactive global illumination techniques, where indirect illumination within a given scene is approximated by a small set of stochastically sampled particles that represent virtual point lights. Shadow maps are commonly exploited on the GPU for interactive rendering to solve the key problem of determining the visibility between particles and visible surface points. Ritschel et al. used precomputed shadow maps [28] for rigid objects, and Laine et al. progressively reused the shadow maps for fast approximation of one-bounce indirect illumination, which eliminated the need for recomputing after every frame [18]. Ritschel et al. approximated visibility with indirect illumination by using low-resolution shadow maps during interactive rendering of moderately complex and fully dynamic scenes [29].

Walter et al. presented a hierarchical, error-bounded solution for the many-light problem, which is similar to the instant radiosity problem, where the visibility is computed based on ray tracing [35,34]. Hašan et al. formulated this problem as a large matrix of sample–light interactions, and applied a shadow mapping method to approximately sample the rows and columns of the matrix [11].

Reflective shadow maps were used by Dachsbacher and Stamminger for fast approximation of one-bounce indirect illumination by ignoring indirect visibility [3, 4]. Multiresolution splatting techniques were applied by Nichols et al. to efficiently compute one-bounce indirect illumination [23,22]. Dachsbacher et al. [5] and Dong et al. [6] took a different approach to the visibility problem by treating visibility

in an implicit manner by transmitting negative radiance. Finally, there are several precomputed radiance transfer methods that support interactive rendering of several interesting global illumination effects at the cost of a nontrivial amount of precomputation, but we omit referencing these here because they are less relevant to our approach. Refer to the work of Sloan et al. [31] and variants.

3 Discretization of light transport space

In this section, we first describe the basic elements that comprise the discretized light transport system, before providing a detailed description of our rendering pipeline in the following section.

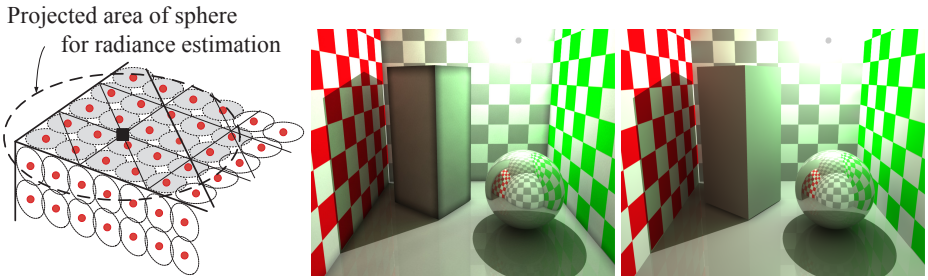
3.1 Area particles and particle links

The most fundamental element in discretization is identifying a set of locations where light interacts with geometries. In our method, we use a collection of *area particles* distributed on the surfaces of geometric models at a given density. Each particle represents a small area around it, which is associated with a list of quantities to describe indirect illumination in the area. These area particles are prepared in a preprocessing stage, to define a basic discrete space where the light transport simulation is performed. They are decoupled from the actual geometries, so light simulation can be conducted independently of the complexities of geometric models.

The next elements of the discretized computational space are light paths between surfaces. One of the most time-consuming components of most global illumination algorithms is the visibility computation, which finds the paths of light propagation. A key motivation behind our work is to reduce the expense of visibility computing by simulating light transport using preselected discrete light paths. Specifically, we construct light paths, called *area particle links* (or just *particle links* for short), that discretize the directional space over the hemisphere around each area particle. A given number of directions are sampled stochastically, based on cosine-weighted hemisphere sampling in a stratified manner. Then, for every direction sample, the method finds the area particle closest to the visible surface point along the direction and associates its ID with the direction. Once this sampling process is completed for all directions and all area particles, we have a collection of particle links of a given resolution to discretely represent the potential light paths in the computational domain.

3.2 Radiance particles and estimation lists

In addition to the visibility computation, another time-consuming component of photon-mapping-style global illumination algorithms is the estimation of the reflected radiances at shading points, which demands repetitive queries to locate the nearest photons around arbitrary surface points during density estimation. To avoid excessive on-the-fly searches (for neighboring area particles in our method) during rendering, we separate the radiance estimation process into two phases.



(a) Implicitly defined radiance particles (b) A result of a *simple* photon mapping based renderer (c) Our result

Fig. 3 Estimation of reflected radiance at radiance particles. Using a preselected set of area particles per radiance particle, a fast estimation of the reflected radiance is possible, which also allows to eliminate the well-known boundary bias problem [13].

First, we estimate diffusively reflected radiance for only a small set of surface points, called *radiance particles*, using preselected area particle lists, called *radiance estimation lists* (or simply *estimation lists* for short). Then, during recursive ray tracing, the indirect illumination at each ray-surface intersection point, whether hit by a primary or secondary ray, is quickly approximated on the fly using a simple linear interpolation of radiance estimates for three surrounding radiance particles.

Decoupling the shading points (i.e., the ray-surface hit points) from the actual locations of reflected radiance estimation makes it possible to avoid any unnecessary, computationally intensive estimation computation without seriously affecting the quality of the global illumination effect. However, identifying the best set of radiance particles is an intractable problem, so our current implementation simply subdivides each triangle of geometric models into subtriangles, by tessellating the three edges such that the resulting elements have sizes smaller than a user-specified threshold value. The vertices of the subtriangles are enumerated in the order of triangle strips to become radiance particles of their respective triangles. There is no need to record the actual coordinates of radiance particles, because information on a subtriangle containing a given shading point can be quickly extracted on the fly during ray tracing.

Once the radiance particles are selected, a list of area particles is constructed for every radiance particle, in the form of a radiance estimation list. Note that, when an estimation list is created, only the area particles that exist within the sphere of a given radius and have similar surface normals to the given radiance particle, are added to the list. It enables to replace expensive searches for neighboring area particles with a low-cost scan of a list. In addition, the estimation list allows a more precise estimation of the actual projected area, simply by adding those of area particles in the list. As a result, we can easily eliminate the problem of boundary biases, which are often produced by photon-mapping style algorithms (see Figure 3).

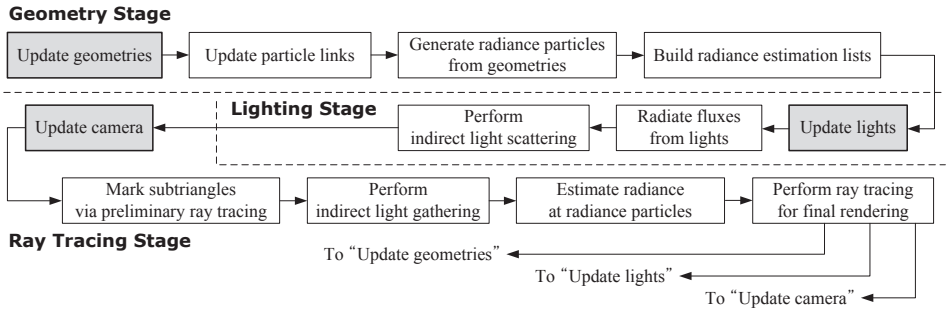


Fig. 4 Rendering pipeline overview.

4 Rendering pipeline

Figure 4 illustrates the system overview of our rendering algorithm, which is split into three phases: *geometry stage*, *lighting stage*, and *ray tracing stage*.

4.1 Geometry stage

When rendering starts, a geometry stage is conducted when necessary; it updates proper acceleration structures to reflect any changes in geometries. First, each link of the area particles is checked for its validity, and is updated if found to be invalid. Next, radiance particles are generated from geometries and a corresponding estimation list is created for each. Unlike the particle link update, this step may be skipped if the geometries are rigid, or only a slight deformation has occurred. Currently, most of the time used during the geometry stage (actually in the entire rendering computation) is spent on the link update, as will be seen, because computationally intensive visibility computation by ray shooting is performed.

4.2 Lighting stage

Responding to dynamic lights and camera, the following two rendering steps rapidly generate global illumination. First, the lighting stage, which is performed whenever there is a change in either geometries or lights, constructs a discrete field of indirect illumination in the area particle space.

4.2.1 Emission from light sources

The first substep in this stage is to distribute light energy from light sources to area particles, by setting their initial fluxes. Unlike the conventional photon mapping algorithm, where the same power of emitted photons is recorded at arbitrary first-hit surface locations, our method distributes the initial energy to preselected particle positions, which requires the use of a modified light emission technique.

In the discrete area particle space, the ith particle is associated with a descriptor that consists of, among other attributes: position $p^{(i)}$, surface normal $\mathbf{n}^{(i)}$, area of influence $\Delta A^{(i)}$, diffuse BRDF $f_{r,d}^{(i)}$ ($= \rho_d^{(i)}/\pi$ for a diffuse reflectance $\rho_d^{(i)}$ at

$p^{(i)}$), flux $\phi^{(i)}$, and reflected radiance $L^{(i)}$. To simulate the direct illumination from the jth point light having a total power Φ_j, we first check the visibility between the light and each area particle (for efficiency, only *front-facing* area particles are examined). When the ith particle is visible, the solid angle of the particle's area of influence when seen from the light is computed as $\frac{\Delta A^{(i)} \cos\theta_i}{d_i^2}$, where d_i is the distance to the light and θ_i is the angle between the surface normal $\mathbf{n}^{(i)}$ and the direction toward the light. The photon power $\frac{\Phi_j}{4\pi} \frac{\Delta A^{(i)} \cos\theta_i}{d_i^2}$ corresponding to the solid angle is then accumulated into the flux $\phi^{(i)}$ of the area particle.

4.2.2 Indirect light scattering via area particles

After completion of the light emission process for each light source, the directly illuminated radiant energy is propagated among the area particles using the particle links. Given a set of particle links for the ith particle with $n_s \times n_s$ cosine-weighted directional samples, one bounce of diffuse reflection on the particle can be approximated by accumulating the evenly divided reflected flux $\frac{\rho_d^{(i)} \cdot \phi^{(i)}}{n_s^2}$ for all the destination particles on the links. Iterating this light bounce for all the area particles would effectively simulate the light propagation process. However, the light scattering operation may generate a considerable number of concurrent writes to the same global memory location, thereby leading to the possibility of inefficiency in the current CUDA computing architecture, as a consequence of repeated uses of the read-modify-write atomic operation.

For efficient light propagation, we first enumerate the particles directly illuminated by lights, and use their full $n_s \times n_s$ links for only the first bounce, which creates the most significant indirect light. In subsequent bounces for each jth end particle, the light is scattered in only one randomly selected link direction, thereby accumulating the reflected energy $\rho_d^{(j)} \cdot \phi^{(j)}$ into the end particle (see Figure 5). Note that the one-direction scattering scheme may introduce subtle high-frequency noises in the scattered indirect illumination. However, they disappear in the final ray tracing stage thanks to the indirect light gathering computation performed at the area particles that participate in shading.

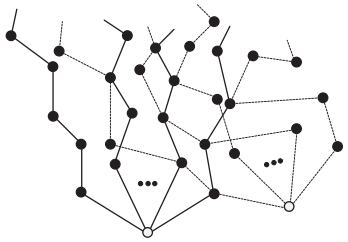


Fig. 5 Indirect light scattering from two area particles. Each computing thread is responsible for light propagation from each directly illuminated area particle, which is simulated by one light bounce with $n_s \times n_s$ direction samples, followed by n_b extra bounces each using a single random direction sample.

When the scattering phase is over, the reflected radiance $L^{(i)}$ for the ith particle, due to both direct and indirect illumination, is estimated using the accumulated flux:

$$L^{(i)} = \int_{\Omega} f_{r,d}^{(i)}\, dE(p^{(i)}, \boldsymbol{\omega}) = f_{r,d}^{(i)} \int_{\Omega} dE(p^{(i)}, \boldsymbol{\omega})$$
$$= f_{r,d}^{(i)} \int_{\Omega} \frac{d^2\Phi(p^{(i)}, \boldsymbol{\omega})}{dA} \approx f_{r,d}^{(i)} \cdot \frac{\sum_k \Delta\Phi(x_k, \boldsymbol{\omega}_k)}{\Delta A^{(i)}},$$

where $\sum_k \Delta\Phi(x_k, \boldsymbol{\omega}_k)$ represents the sum of fluxes that have arrived in the ith particle's area. Since this is exactly $\phi^{(i)}$, $L^{(i)} \approx f_{r,d}^{(i)} \cdot \frac{\phi^{(i)}}{\Delta A^{(i)}}$.

4.3 Ray tracing stage

The rendering computation is completed as follows. First, preliminary ray tracing is performed without shading or shadow computations to mark the subtriangles hit by primary and/or secondary rays, thereby extracting a list of radiance particles to be used for shading. Once the area particles found in the estimation lists of the extracted radiance particles are collected, the indirect light gathering process is performed for only those area particles. Radiance is then estimated for the extracted radiance particles using their estimation lists based on the indirect light field in the area particle space. Finally, the radiance field in the radiance particle space is used in the final recursive ray tracing stage, to add the indirect illumination effect.

4.3.1 Indirect light gathering at area particles

The forward light scattering mechanism we describe here results from an effort to find a balance between computational time and quality of light propagation simulation. This method is efficient, but it may result in visual artifacts, such as blotchiness, which is often caused by insufficient sampling of light paths. To alleviate this problem, an additional pass of light transport, known as *indirect light gathering*, is conducted only for the relevant area particles to refine the discrete flux field by collecting incoming radiance from other particles. This extra step results in improved quality of global illumination, which compares favorably to that achieved with the final gathering method. More importantly, this additional gathering step ensures that the area particles hold the correct indirect illumination due to diffuse interreflection, whereas the accumulated flux from the indirect light scattering step contains light energy from both direct and indirect illumination.

In our method, we again use the particle links, which are generated based on a cosine-weighted hemisphere sampling with PDF $P(\boldsymbol{\omega}) = \frac{\cos\theta}{\pi}$ to approximate the refined irradiance $\tilde{E}^{(i)}$ at the ith particle using a Monte Carlo estimator:

$$\tilde{E}^{(i)} = \int_{\Omega} L_{in}(p^{(i)}, \boldsymbol{\omega})\, \cos\theta\, d\boldsymbol{\omega}$$
$$\approx \frac{1}{N_{gather}} \sum_{k=1}^{N_{gather}} \frac{L_{in}(p^{(i)}, \boldsymbol{\omega}_k) \cos\theta_k}{P(\boldsymbol{\omega}_k)}$$

$$= \frac{\pi}{N_{gather}} \sum_{k=1}^{N_{gather}} L_{in}(p^{(i)}, \boldsymbol{\omega}_k),$$

where N_{gather} is the number of direction samples of the particle links and θ_k is the angle between the kth sample direction $\boldsymbol{\omega}_k$ and the normal at $p^{(i)}$. To approximate the incoming radiance $L_{in}(p^{(i)}, \boldsymbol{\omega}_k)$ from the direction $\boldsymbol{\omega}_k$, let $p^{(k^*)}$ be the end particle corresponding to the direction. Then, under the assumption that the actual intersection point of the kth gather ray is close enough to $p^{(k^*)}$, we have an approximation $L_{in}(p^{(i)}, \boldsymbol{\omega}_k) \approx L(p^{(k^*)}, -\boldsymbol{\omega}_k) \approx L^{(k^*)}$. Thus, using the light flux accumulated in the indirect light scattering step, we obtain an improved irradiance estimate at $p^{(i)}$:

$$\tilde{E}^{(i)} \approx \frac{\pi}{N_{gather}} \sum_{k=1}^{N_{gather}} f_{r,d}^{(k^*)} \cdot \frac{\phi^{(k^*)}}{\Delta A^{(k^*)}}.$$

This expression is simplified when area particles are distributed uniformly with constant area ΔA:

$$\tilde{E}^{(i)} \approx \Big(\frac{\pi}{N_{gather}\Delta A}\Big) \sum_{k=1}^{N_{gather}} f_{r,d}^{(k^*)} \cdot \phi^{(k^*)},$$

which leads to more efficient CUDA implementation. By approximating the new flux at the ith particle as $\tilde{\phi}^{(i)} \approx \tilde{E}^{(i)} \cdot \Delta A^{(i)}$, we arrive at a well-smoothed indirect light field in area particle space.

4.3.2 Radiance estimation at radiance particles

After the gathering computation, the radiance $\bar{L}^{(l)}$ at the lth radiance particle, $q^{(l)}$, located on a surface with diffuse BRDF $f_{r,d}^{(l)}$, is estimated using the area particles in its estimation list, as follows:

$$\bar{L}^{(l)} = f_{r,d}^{(l)} \cdot \frac{d\Phi(q^{(l)})}{dA} \approx f_{r,d}^{(l)} \frac{\sum_m \tilde{\phi}^{(m)}}{\sum_m \Delta A^{(m)}}.$$

Again, this becomes $\bar{L}^{(l)} \approx \frac{f_{r,d}^{(l)}}{m\Delta A} \sum_m \tilde{\phi}^{(m)}$ when the area particles are distributed uniformly.

4.3.3 Final ray tracing with adaptive supersampling

In the final substep, a recursive ray tracer is run where, for every ray-surface hit along each ray path, the linearly interpolated indirect color is combined with those from direct illumination and specular reflection/refraction. In addition to a single pixel sampling version, we implemented an augmented ray tracer for producing high-quality rendering images, performing antialiasing based on the recent adaptive supersampling method [14], which is known to support high sampling rates as effective as 9 to 16 per pixel. The experimental results for the two ray tracers are given next.

5 Implementation results

To show the effectiveness of our method, we implemented the extended ray tracer using the CUDA platform [24], and tested performance on an NVidia GeForce GTX 680 GPU. Figure 2 (BATHROOM) and Figure 6 (CBOX, SPONZA, KITCHEN, CONFERENCE, and MAZE) show six selected test scenes, which were ray-traced at a resolution of 1024×1024 with shading, textures, reflection/refraction, and shadows using appropriate numbers of area and radiance particles.

5.1 Computation time and memory space

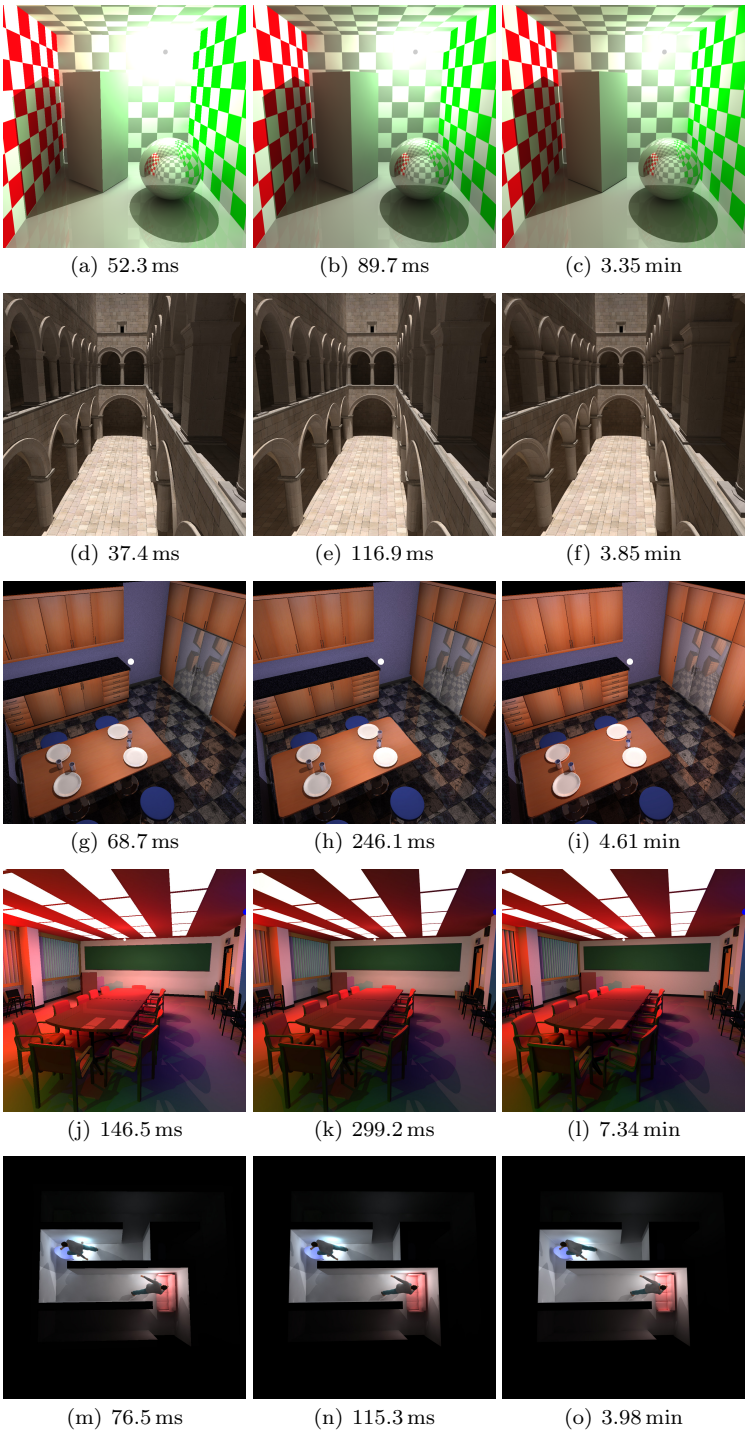
As shown in Table 1, most of the time allocated to the geometry stage was consumed by the construction of the 16×16 particle links required for indirect light gathering, whereas building the radiance estimation lists cost relatively little time. The memory space required for storing area particles was roughly proportional to the number of distributed particles. In contrast, the size of the radiance estimation lists depended on both the number of generated radiance particles and the radius of the radiance estimation. Our current implementation builds a radiance estimation list for every generated radiance particle, but the memory requirement would decrease markedly if these were constructed only for the radiance particles that are actually used, as detected in the ray tracing stage. Furthermore, the entire area particle links are newly built by shooting rays from all area particles during visibility computation. Thus, if only part of the geometries had changed, then considerably fewer actual links would require updating. Application of an optimization technique to link updating will be addressed in a future implementation.

(Time unit: second)

	CBOX	SPON	KITC	CONF	BATH	MAZE
Triangles	8,024	66,454	98,503	190,947	256,907	241,080
Area particles	50,000	150,000	150,000	200,000	200,000	150,000
4×4 scattering link generation	0.08 (3.2MB)	0.26 (9.5MB)	0.38 (9.5MB)	1.15 (12.9MB)	0.6 (12.9MB)	0.4 (9.5MB)
16×16 gathering link generation	1.10 (48.8MB)	3.24 (146.5MB)	4.75 (146.5MB)	15.09 (195.3MB)	7.79 (195.3MB)	4.92 (146.5MB)
Tessellation [Gen. rad. particles]	0.04 [51,348]	0.39 [596,914]	0.40 [597,362]	0.53 [812,199]	1.09 [1,713,813]	0.72 [1,119,435]
Rad. estimation list generation	0.02 (10.3MB)	0.15 (120.1MB)	0.19 (135.7MB)	0.50 (367.4MB)	0.58 (215.7MB)	0.39 (365.9MB)

Table 1 Analysis of the geometry stage.

The two tables shown in Tables 2(a) and (b) indicate the actual rendering times with and without application of the adaptive supersampling technique [14]. In this experiment, we selected the number of light scattering bounces (n_b) by incrementally increasing it until using a higher bounce made hardly any visual difference with our method (note that the actual light bounces handled in the rendering is $n_b + 1$ including the indirect light gathering step).



(a) 52.3 ms (b) 89.7 ms (c) 3.35 min

(d) 37.4 ms (e) 116.9 ms (f) 3.85 min

(g) 68.7 ms (h) 246.1 ms (i) 4.61 min

(j) 146.5 ms (k) 299.2 ms (l) 7.34 min

(m) 76.5 ms (n) 115.3 ms (o) 3.98 min

Fig. 6 Rendering results. The first and second images in each triple show our ray-traced results using single pixel sampling and the adaptive supersampling technique [14], respectively. These are compared with the third image, generated using an optimized GPU ray tracer that fully supports the traditional photon mapping/final gathering method.

(a) Rendering time with single pixel sampling (ours)

	CBOX	SPON.	KITC.	CONF.	BATH.	MAZE
Lights	1	1	1	4	2	2
Scattering bounces (n_b)	8	8	4	3	7	13
Total time (ms)	52.2	37.4	68.7	146.5	146.8	76.5
[FPS]	[19.14]	[26.72]	[14.56]	[6.82]	[6.81]	[13.08]
Light emission	0.2%	0.2%	0.1%	0.1%	0.1%	0.1%
n_b-bounce scattering	1.5%	3.5%	2.4%	8.0%	4.3%	4.6%
Prelim. ray tracing	50.6%	30.5%	37.9%	33.8%	29.8%	46.6%
[Found rad. particles]	[35,844]	[76,362]	[130,533]	[56,979]	[212,866]	[142,661]
16×16 gathering	1.8%	4.0%	4.8%	2.4%	3.0%	4.7%
[Used area particles]	[43,170]	[23,232]	[84,257]	[96,645]	[88,194]	[73,297]
Radiance estimation	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Ray tracing	45.8%	61.7%	54.7%	55.6%	62.7%	43.9%

(b) Rendering time with adaptive supersampling [14] (ours)

	CBOX	SPON.	KITC.	CONF.	BATH.	MAZE
Total time (ms)	89.7	116.8	246.3	299.4	293.3	115.3
[FPS]	[11.15]	[8.56]	[4.06]	[3.34]	[3.41]	[8.67]
Steps in table (a)	58.3%	32.0%	27.9%	49.0%	50.1%	66.3%
Alias detection	8.3%	7.1%	3.8%	2.6%	2.7%	6.3%
Prelim. ray tracing†	9.5%	17.3%	18.9%	9.1%	10.4%	4.4%
[Add. rad. particles]	[512]	[21,366]	[78,068]	[17,155]	[137,245]	[50,547]
16×16 gathering†	3.2%	5.8%	5.3%	2.8%	4.0%	6.6%
[Add. area particles]	[310]	[726]	[13,429]	[3,296]	[4,243]	[421]
Radiance estimation†	0.3%	0.7%	1.0%	0.8%	1.6%	2.0%
Adaptive ray tracing	20.4%	37.1%	43.1%	35.7%	31.2%	14.4%

(c) Rendering time (Whitted style ray tracing)

	CBOX	SPON.	KITC.	CONF.	BATH.	MAZE
Single pixel sampling (FPS)	40.8	43.2	36.6	18.4	15.5	42.3
[Ours/Whitted]	[0.469]	[0.619]	[0.398]	[0.371]	[0.439]	[0.309]
Adaptive pixel sampling (FPS)	18.0	16.9	7.9	5.7	5.4	19.0
[Ours/Whitted]	[0.619]	[0.507]	[0.514]	[0.586]	[0.631]	[0.456]

Table 2 Analysis of the lighting and ray tracing stages.

The frame rates achieved for ray tracing 1024×1024 images (6.81 to 26.72 fps for single pixel sampling and 3.34 to 11.15 fps for adaptive supersampling) were affected by several parameters, including the complexities of geometries, lighting conditions, and material properties. In many cases, the time spent on the ray tracing process usually ranged from 40% to 60% of the total rendering time, regardless of whether adaptive sampling was applied; this suggests that the additional cost of including global illumination is roughly the same on average as that found with basic Whitted-style ray tracing. This fact was confirmed by the timings shown in Table 2(c), which indicate the frame rates measured in the same environment for a pure Whitted ray tracer.

Detailed analysis of the relative overheads in the single sampling case (Table 2(a)) found that most of the global illumination time was usually consumed in the processes of indirect light scattering, gathering, and preliminary ray tracing for finding the radiance particles that participate in shading. As with other stages,

the scattering step was affected to a high degree by the lighting condition. For instance, the KITCHEN scene, rendered with 150,000 area particles, required 1.65 ms for four-bounce light scattering, whereas the SPONZA scene, rendered with the same number of area particles, took only 1.31 ms for eight-bounce scattering. The reason for this difference was that direct illumination from the sky in the SPONZA scene illuminated a relatively lower number of area particles, thereby generating fewer computing threads in the scattering stage. In such a situation, this forward light transport is more efficient than repeated backward gathering per light bounce. Previous researchers, such as [30], simulated light propagation by iterating the gathering operation over all vertices. This works well in a situation where scene geometries are directly well illuminated, but an exclusively backward gathering approach might often result in inefficiency, because it requires memory access to gather radiance from the locations that receive little or no radiance. Furthermore, processing the gathering operation for all elements for every bounce could generate a significant workload when multiple bounces of light propagation are required for nontrivial scenes. In contrast, our method minimizes the expensive light gathering operation in a view-dependent manner, by performing this process only for area particles in the vicinity of recursively spawned ray paths. In the SPONZA scene, this was conducted only for 23,232 area particles (only 15.5% of the distributed particles), which took 1.50 ms, whereas 3.30 ms was required for the light gathering for 84,257 area particles (56.2%) in the KITCHEN scene, where a greater area was passed directly or indirectly by recursive rays (see also Table 3).

(Time unit: millisecond)

	CBOX	SPON.	KITC.	CONF.	BATH.	MAZE
Gathering only	56.3	84.3	87.1	230.8	139.1	112.3
Scattering/gathering	24.6	3.6	13.1	29.5	21.4	27.4

Table 3 Comparison with a gathering-only method. To evaluate the efficiency of the proposed light transport scheme, we measured on an NVidia GeForce GTX 580 GPU the timings for repeating the gathering operation three times, as in [30], at all area particles with 256 links each (Gathering only). The effectiveness of our two-way approach, through which higher-quality global illumination is generated, is clearly observed in the Scattering/gathering row which shows the timings for carrying out the n_b-bounce scattering and 16×16 gathering.

On the other hand, applying the adaptive supersampling technique incurred an extra cost for global illumination, marked by † in Table 2(b), which was mostly spent in the preliminary ray tracing step, locating all the radiance particles, and thus the area particles, along the additional ray paths required for adaptive supersampling. In contrast, the cost of additional light gathering and radiance estimation was found to be very low.

5.2 Image quality

We analyzed the quality of the global illumination generated using our method by implementing a full ray tracer supporting photon mapping and final gathering on the GPU, where 4,096 gather rays were shot to produce ground truth images (see

Figures 2 and 6). The rendering results indicate that our method produced global illumination that compared favorably to the reference images, which was often difficult to find visual differences when rendered interactively. A side effect of using preselected locations for light scattering and gathering in particle spaces was that it created very few temporal aliases for dynamic lights and camera.

Like all numerical approximation methods, ours is not free of discretization errors, particularly in the 'high-frequency' regions. The different images in Figure 7(a) and (b) show visual errors obtained when rendering the two example scenes using 200,000 area particles, and illustrate a typical pattern of discrepancy between the reference images and our results. The most prominent differences are found in the surface regions, such as the corners, where indirect illumination varies rapidly, which is mainly due to insufficient area particles. Moreover, our light emission method, which differs from the standard photon mapping that was applied to make the reference images, also produced a slight difference, especially when lights were very close to surfaces, as shown in the ceiling of the CONFERENCE scene and the upper right corners of the two images in Figures 6(a) to (c). We suggest that the two light emission techniques created slightly different light attenuation effects on the closest surfaces, but they hardly had a significant influence on the appearance of diffuse interreflection in general. To discern how a more refined discretization reduces errors in the 'high-frequency' regions, we incrementally increased the number of area particles. As shown in the two sets of examples in Figure 7(c) to (f) and (g) to (j), the errors gradually decreased at the cost of lower frame rates. In these tests, we distributed the area particles uniformly in the entire scenes. A multilevel adaptive distribution of particles would efficiently decrease visual errors.

6 Concluding remarks

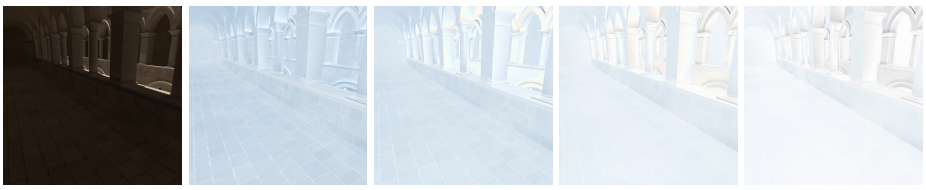
As we have demonstrated, our method efficiently handles dynamic lights and viewpoints to generate a global illumination effect that compares favorably to the traditional final gathering method. It can be used as a fast relighting tool for architectural and cinematic lighting. However, the current method is not interactive for fully dynamic scenes, because geometric updating of the particle spaces in response to a changed geometry requires several seconds. It remains for future research to develop an efficient algorithm for area particle link updating, which would be optimized for specific animation requirements. In particular, for situations such as the maze scene illustrated in Figure 1, where two people move around the maze, only the area particle links that are adjacent to or intersected by a moving person need to be modified. For example, we believe that an acceleration structure similar to that used in the line-space hierarchy method for radiosity [7, 10] could allow interactive updating in our discrete geometry system.

Note that the global illumination information obtained in the lighting stage could also be exploited easily for shading by the pixel shader in a rendering pipeline based on OpenGL/DirectX. As shown in the previous section, the ray tracing stage that actually performs the final shading computation usually takes 40–60% of the rendering time after the geometry stage (see Table 2(a)). By replacing this time-consuming element with a computationally cheaper OpenGL/DirectX-based renderer, we could develop a faster alternative rendering method that would support

(a) CONF. (b) BATH.

(c) Cropped FG image (BATH.) (d) 200K area particles at 6.5 fps (e) 400K area particles at 4.7 fps (f) 600K area particles at 2.8 fps

(g) Another FG image (CONF.) (h) 200K area particles at 5.9 fps (i) 400K area particles at 3.9 fps (j) 600K area particles at 3.1 fps



(k) SPON. (FG) (l) 1 scattering bounce at 12.410 fps (2x difference) (m) 3 scattering bounces at 12.361 fps (2x difference) (n) 6 scattering bounces at 12.338 fps (2x difference) (o) 9 scattering bounces at 12.285 fps (2x difference)

Fig. 7 Comparison with final gathered images. Here, the error images are displayed in negative colors to show the differences more clearly. The frame rates indicate the times taken in the lighting and ray tracing stages to generate 1024×1024 adaptively supersampled images. The error images in (a) and (b) obtained from the presented results indicate that most of differences were usually concentrated in 'high-frequency' regions, where indirect illumination changes relatively faster. In another view of the BATHROOM scene, we incrementally increased the number of area particles in the scene and found that the differences in the corners decreased gradually ((c) to (f)). The same phenomenon was observed in another view of the CONFERENCE scene ((g) to (j)), which strongly suggests the use of a multilevel adaptive particle distribution method. In a poor lighting condition, insufficient indirect light bounces often generate too dark indirect illumination. In the example shown in (k), we had to increase the number of scattering bounces up to 8 or 9, for which our light scattering/gathering scheme did not cause a marked performance drop ((l) to (o)).

effective global illumination. Finally, we are currently exploring an effective multi-level algorithm for detecting high-frequency surface regions rapidly and adaptively, to which additional area particles could be distributed. This will require applying some variant of a criterion such as the harmonic mean distance, which is used in irradiance caching [38].

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MOE) (No. 2012R1A1A2008958).

References

1. O. Arikan, D. Forsyth, and J. O'Brien. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Transactions on Graphics*, 24(3):1108–1114, 2005.
2. P. Christensen. Point-based approximate color bleeding. Pixar Technical Memo 08–01, 2008.
3. C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *Proc. of the Symposium on Interactive 3D Graphics and Games*, pages 203–231, 2005.
4. C. Dachsbacher and M. Stamminger. Splatting indirect illumination. In *Proc. of the Symposium on Interactive 3D Graphics and Games*, pages 93–100, 2006.
5. C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics*, 26, 2007. Article No. 61.
6. Z. Dong, J. Kautz, C. Theobalt, and H.-P. Seidel. Interactive global illumination using implicit visibility. In *Proc. of Pacific Graphics*, pages 77–86, 2007.
7. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proc. of the SIGGRAPH '97*, pages 57–64, 1997.
8. B. Fabianowski and J. Dingliana. Interactive global photon mapping. *Computer Graphics Forum*, 28(4):1151–1159, 2009.
9. P. Gautron, J. Křivánek, K. Bouatouch, and S. Pattanaik. Radiance cache splatting: a GPU-friendly global illumination algorithm. In *Proc. of the Eurographics Symposium on Rendering*, pages 55–64, 2005.
10. X. Granier and G. Drettakis. Incremental updates for rapid glossy global illumination. *Computer Graphics Forum*, 20(3):268–277, 2001.
11. M. Hašan, F. Pellacini, and K. Bala. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics*, 26, 2007. Article No. 26.
12. H. W. Jensen. Global illumination using photon maps. In *Proc. of the Eurographics Workshop on Rendering Techniques*, pages 21–30, 1996.
13. H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Ltd., ISBN 1-56881-147-0, 2001.
14. B. Jin, I. Ihm, B. Chang, C. Park, W. Lee, and S. Jung. Selective and adaptive supersampling for real-time ray tracing. In *Proc. of High Performance Graphics*, pages 117–125, 2009.
15. J. Kajiya. The rendering equation. In *Proc. of ACM SIGGRAPH*, volume 20, pages 143–150, 1986.
16. A. Keller. Instant radiosity. In *Proc. of ACM SIGGRAPH*, pages 49–56, 1997.
17. J. Křivánek, P. Gautron, S. Pattanaik, and K. Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005.
18. S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila. Incremental instant radiosity for real-time indirect illumination. In *Proc. of the Eurographics Symposium on Rendering*, pages 277–286, 2007.
19. B. Larsen and N. Christensen. Simulating photon mapping for real-time applications. In *Proc. of the Eurographics Symposium on Rendering*, pages 123–131, 2004.
20. J. Lehtinen, M. Zwicker, E. Turquin, J. Kontkanen, F. Durand, F. Sillion, and T. Aila. A meshless hierarchical representation for light transport. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2008)*, 27, 2008. Article No. 37.
21. M. McGuire and David D. Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proc. of High Performance Graphics*, pages 77–89, 2009.

22. G. Nichols, J. Shopf, and C. Wyman. Hierarchical image-space radiosity for interactive global illumination. *Computer Graphics Forum*, 28(4):1141–1149, 2009.
23. G. Nichols and C. Wyman. Multiresolution splatting for indirect illumination. In *Proc. of the Symposium on Interactive 3D Graphics and Games*, pages 83–90, 2009.
24. NVIDIA. *NVIDIA CUDA: NVIDIA CUDA C Programming Guide (Version 5.0)*, 2012.
25. T. Purcell, C. Donner, M. Cammarano, H. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proc. of Graphics Hardware*, pages 41–50, 2003.
26. T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012.
27. T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics*, 28(5), 2009. Article No. 132.
28. T. Ritschel, T. Grosch, J. Kautz, and H.-P. Seidel. Interactive global illumination based on coherent surface shadow maps. In *Proc. of Graphics Interface*, pages 185–192, 2008.
29. T. Ritschel, T. Grosch, M. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics*, 27, 2008. Article No. 129.
30. A. Schmitz, M. Tavenrath, and L. Kobbelt. Interactive global illumination for deformable geometry in CUDA. *Computer Graphics Forum*, 27(7):1979–1986, 2008.
31. P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21:527–536, 2002.
32. E. Tabellion and A. Lamorlette. An approximate global illumination system for computer generated films. *ACM Transactions on Graphics*, 23(3):469–476, 2004.
33. T. Umenhoffer and L. Szirmay-Kalos. Robust diffuse final gathering on the GPU. In *Proc. of WSCG*, 2007.
34. B. Walter, A. Arbree, K. Bala, and D. Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics*, 25(3):1081–1088, 2006.
35. B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics*, 24(3):1098–1107, 2005.
36. R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao. An efficient GPU-based approach for interactive global illumination. *ACM Transactions on Graphics*, 28, 2009. Article No. 91.
37. G. Ward and P. Heckbert. Irradiance gradients. In *Proc. of the Eurographics Workshop on Rendering*, pages 85–98, 1992.
38. G. Ward, F. Rubinstein, and R. Clear. A ray tracing solution for diffuse interreflection. In *Proc. of ACM SIGGRAPH*, pages 85–92, 1988.
39. C. Yao, B. Wang, B. Chan, J. Yong, and J.-C. Paul. Multi-image based photon tracing for interactive global illumination of dynamic scenes. *Computer Graphics Forum*, 29:1315–1324, 2010.
40. K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time KD-tree construction on graphics hardware. *ACM Transactions on Graphics*, 27(5):1–11, 2008.