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

Ray Tracing-based Interactive Diffuse Indirect Illumination

Byungjoon Chang · Sanghun Park · Insung Ihm

Received: date / Accepted: date

Abstract Despite great efforts in recent years to accelerate global illumination computation, the real-time ray tracing of fully dynamic scenes to support photo-realistic indirect illumination effects has yet to be achieved in computer graphics. In this paper, we propose an extended ray tracing model that can be readily implemented on a GPU to facilitate the interactive generation of diffuse indirect illumination, the quality of which is comparable to that generated by the traditional, time-consuming photon mapping method and final gathering. Our method employs three types of (multilevel) grids to represent the indirect light in a scene using a form that facilitates the efficient estimation of the reflected radiance caused by diffuse interreflection. This method includes the mathematical tool of spherical harmonics and a rendering scheme that performs the final gathering step with a minimal cost during ray tracing, which guarantees the interactive frame rates. We evaluated our technique using several dynamic scenes with nontrivial complexity, which demonstrated its effectiveness.

Keywords ray tracing · global illumination · dynamic scenes · diffuse interreflection · final gathering · grids · spherical harmonics

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

Classical ray tracing was introduced to the graphics community by Whitted [33] and it has been used frequently for the high-quality rendering of virtual scenes. Whitted-style ray tracing is particularly well suited to the generation of specular shading effects such as reflection and refraction, for which the rasterization method does not provide an exact solution. Like other rendering methods, however, the application of ray tracing to the precise real-time reproduction of the optical phenomenon of *diffuse indirect illumination* remains a major challenge, because it involves the reflection of incident light from diffuse surfaces, possibly after multiple reflections from other diffuse surfaces.

Stochastic path tracing is effective for producing such diffusive reflection effects, but its computational overheads are too high for interactive rendering in practice. A practical approach to diffuse indirect illumination in the classical ray tracing scheme is photon mapping [8], where the diffusive phenomenon is simulated by tracing global photons in the scene and estimating the reflected radiances using photons stored in the vicinity of shading points. The rendering quality of photon mapping is improved greatly by final gathering [32], which samples the incident radiance at each shading point in a stochastic manner before removing any visually objectionable blotchiness that might be caused by the direct visualization of insufficient photon samples.

A combination of Whitted-style ray tracing, photon mapping, and final gathering has been used routinely by the special effects industry to synthesize high-quality images. However, this hybrid method barely achieves an interactive frame rate in nontrivial scenes when implemented according to the original design because of the large amounts of visibility computation required for photon tracing and radiance gathering, as well as the repetitive photon queries needed for radiance estimation. Although there has been substantial recent research into the development of practical algorithms, the reproduction of high-quality diffuse interreflection in an interactive manner in complex scenes remains a difficult problem using this combined rendering scheme.

1.2 Our contribution

In this study, we propose a novel interactive method, which can be integrated readily into the conventional Whitted-style ray tracer to simulate diffuse indirect illumination in an effective manner. Our rendering algorithm aims to create a global illumination effect rapidly for fully dynamic scenes, and to achieve a quality comparable to that generated using the computationally intensive photon mapping and final gathering techniques. To develop a GPU-friendly light transport model, we discretize the scene domain using three grids, as follows. A *voxelization grid* provides accuracy and efficiency during the visibility computation required to sample reflected radiance. A *multilevel radiance grid* represents the reflected radiance attributable to indirect light in a multi-resolution form, thereby facilitating effective radiance filtering. Finally, a *spherical harmonics grid* provides a computational domain where a *spherical harmonics field* is constructed for the indirect

light in the vicinity of the surfaces of geometric models, which allows efficient final gathering during the ray tracing stage.

Our method extends the Whitted-style ray tracing model so it naturally inherits the advantage of recursive ray tracing, which is inherently lacking in the rasterization method. It implements photon mapping in a basic manner. Therefore, as well as rendering effects such as caustics, the presented method allows the efficient generation of diffuse interreflection that often requires many indirect light reflections, which many previous interactive techniques cannot handle. Finally, our computational scheme effectively simulates final gathering to synthesize high-quality images, which also creates very few temporal aliases for dynamic scene changes despite using only a small number of gather rays. The comparable interactive global illumination method by Wang et al. [30] employs the illumination-cut and shading-cluster techniques to reduce the excessive computational cost of the conventional photon mapping and final gathering. On the other hand, our method attempts to achieve the rendering performance both in terms of computation speed and global illumination quality by implementing these advanced global illumination methods in the grid spaces that effectively approximate both geometry and light transport.

2 Previous work

There is a vast body of research on interactive global illumination, so we only consider the most relevant work in this section (for an extensive survey of this topic, refer to [19]). During the development of an interactive ray tracer based on photon mapping and final gathering, there is a bottleneck when large numbers of rays are sent out to gather the incoming radiances. Several caching and interpolation techniques have been proposed to reduce the cost of this process by performing the final gathering at a sparse set of surface points only [32, 31, 26, 11, 1]. These methods are very effective on a CPU, but they are not well suited to efficient GPU implementation because of their inherent computing structures, which usually require sequential queries and cache updates. By contrast, the first GPU implementation of photon mapping by Purcell et al. [18] and its variants greatly accelerated photon mapping and final gathering on the GPU (for some examples of early work, refer to [12, 5, 28]).

Zhou et al. developed a GPU-assisted kd-tree algorithm and used it to implement a fast photon mapping-based ray tracer [35]. 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 the centers of carefully selected surface point clusters [30]. Fabianowski and Dingliana extended the concept of photon differentials to the interactive handling of diffuse reflections for dynamic lights and cameras [4]. The micro-rendering approach proposed by Ritschel et al. [20] is a surfel-based global illumination method that performs final gathering by rasterizing a point-based representation of the scene using a multi-resolution scheme, which is similar to the tree structures used in [29]. Image space photon mapping methods were also proposed by McGuire and Luebke [15] and Yao et al. [34], which facilitate interactive global illumination in the context of rasterization-based rendering pipelines. Maletz and Wang proposed

a method for GPU-based final gathering, which uses importance point projection instead of traversing a point hierarchy [13].

Spherical harmonics are an efficient means of approximating low-frequency functions and they may provide an effective tool for representing slowly varying global illumination (in the context of computer graphics, refer to introductory articles [6, 22]). Several successful techniques for approximating indirect light in static scenes using precomputed spherical harmonics functions have been introduced [25, 24, 10, 11]. To overcome the limitation of precomputed radiance transfer support only in static scenes, Iwasaki et al. proposed a method that is suitable for objects with restricted movement [7]. Sloan et al. developed a technique that reduced the number of spherical harmonics samples required for real-time global illumination computation [23].

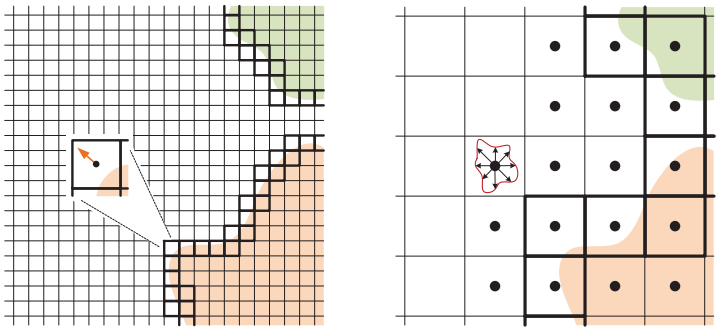
Grid-based techniques were also utilized for faster computation of global illumination mostly within the rasterization scheme. Nijasure et al. rendered the cube map on the GPU to build an incoming radiance field using spherical harmonics in a low-resolution grid, and shaded surface points via interpolation of spherical functions from neighboring grid points [16]. Kaplanyan and Dachsbacher produced real-time indirect illumination with a light propagation volume obtained through simple successive local iterations [9]. Mavridis and Papaioannou utilized the raterization pipeline to approximate indirect light using a volume grid initialized from a directly lit point cloud [14]. In stead of the cube map, Papaioannou used the reflective shadow map [3] for faster sampling of indirect light at grid points, and applied a radiance hint technique to evaluate surface irradiance on nearby surfaces and approximate secondary diffuse interreflections [17]. Thiedemann et al. developed an atlas-based voxelization method and an enhanced ray-voxel intersection technique that allowed real-time near-field illumination and interactive global illumination computations [27]. Crassin et al. presented a hierarchical representation of incoming radiance in a grid space where a cone tracing technique enabled efficient computation of indirect illumination [2].

3 Rendering algorithm: the preparation stage

Our rendering algorithm proceeds in two stages. In the *preparation stage*, a spatial data structure that accelerates ray–object intersections is rebuilt or updated to reflect the changes in geometry between frames. Next, an additional data structure called a *spherical harmonics field* is constructed in the vicinity of the geometry to model the problematic diffuse indirect illumination, which results from diffusive interreflection between objects. In the subsequent *ray tracing stage*, a slightly extended Whitted-style ray tracer is executed to produce a final rendering image with the prepared data structures. This section begins with a step-by-step explanation of how the indirect light is captured in the form of a spherical harmonics field.

3.1 Construction of two-level volumetric grids

The first step when building the spherical harmonics field is the definition of a two-level volumetric grid structure, which comprises a *spherical harmonics grid (SH-*



(a) Voxelization grid (b) Spherical harmonics grid

Fig. 1 Two-level three-dimensional volume grids. The SH-grid defines where the diffuse indirect illumination is approximated using spherical harmonics. The finer, V-grid provides detailed geometry information for voxelized objects.

grid) and a *voxelization grid (V-grid)*, where the SH-grid aims to represent the diffuse indirect illumination in a basic manner whereas the finer, V-grid provides information on voxelized objects in the scene, thereby facilitating a more precise estimation of global illumination in the later stage. For frames that require geometry updates, a GPU-assisted surface voxelization method is applied on the fly to rebuild the V-grid, which marks the *boundary cells* that intersect with triangles. During this process, the triangle indices are not sorted into these cells, but the average normals of the intersecting triangles are retained in the memory (see Figure 1(a)). The SH-grid is produced simply by merging every set of $4 \times 4 \times 4$ V-grid cells into a single cell, which is classified as a boundary cell only if at least one of the corresponding V-grid cells is on the boundary (see Figure 1(b)). The centers of the new boundary cells and their 26 neighbors are then treated as special locations, for which the spherical functions of incoming indirect light are approximated later. Note that the normal information of the V-grid is not inherited by the SH-grid.

3.2 Construction of a global photon list

After the two-level grid structure has been prepared, photon tracing is performed with some slight modifications to the traditional photon mapping technique [8]. First, a global photon that hits a diffusive surface is stored in a *global photon list* only if it is determined to be reflected in a stochastic manner. Second, in contrast to the conventional method where the incoming direction of the photon is usually stored, the reflected, outgoing direction, which is also selected stochastically, is recorded with the photon's RGB-valued radiant flux. Third, the index of the cell in the SH-grid that contains the hit location is also stored with the photon so the list of global photons can be sorted in an appropriate manner to facilitate efficient GPU computation (in our implementation, a cell is referred to based on the index (p, q, r) of its corner voxel with the smallest coordinate values). The array marked **[A]** in Figure 3 shows part of the global photon list, where the indices of the stored photons are shown coupled with triples that indicate their cell indices.

3.3 Construction of multilevel radiance grids

The next step involves the construction of an intermediate multilevel grid structure from the global photon list, referred to as a *radiance grid (R-grid)*, where the base grid has the same configuration as the SH-grid. During this process, the diffuse indirect illumination, which is captured in terms of the radiant fluxes of global photons, is transformed into the radiance and stored on the faces of the cells in the base grid.

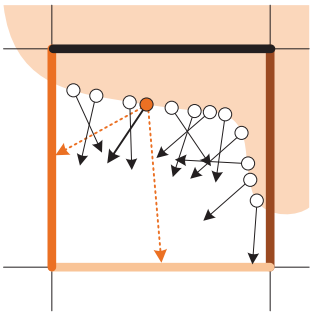


Fig. 2 Distribution of photon fluxes on the 'visible' cell faces.

Consider an ith global photon in a cell with a photon power Φ_i and an outgoing direction $(d_{i_x}, d_{i_y}, d_{i_z})$ (see Figure 2). Multiplied with the bidirectional reflectance distribution function (BRDF) f_{r,d_i} and the respective weight δ_i ($= |d_{x_i}|^2$, $|d_{y_i}|^2$, or $|d_{z_i}|^2$), the photon power is distributed to each of up to three faces of the cell pointed to by the outgoing direction (note that the weights correspond to the squares of the inner product between the outgoing direction and the respective face normals, for which the sum is one). After iterating this calculation for all the global photons, accumulating their weighted radiant fluxes for the appropriate faces, and dividing the outcomes by the face area A, we obtain an indirect light field where the quantity associated with each directional face of a cell, i.e., $\sum_i f_{r,d_i} \frac{\delta_i \Phi_i}{A} = \sum_i f_{r,d_i} \Delta E_i$ for the differential irradiance ΔE_i, approximates the reflected radiance attributable to diffuse indirect illumination, which emanates from the surface inside the cell through the face.

A simple method for implementing the accumulation process on the GPU would be to map each global photon to a GPU thread, which then performs the respective accumulation calculation. This is easy to code but such a scheme would be highly reliant on expensive atomic add operations if parallel threads are used to accumulate the photon powers to identical faces simultaneously. In our implementation, we use the more complex, but experimentally more efficient GPU algorithm, shown in Figure 3, the computational steps of which are as follows. First, the aforementioned index array (marked as **[A]**) is sorted in parallel based on the cell indices, which places photons in the same boundary cell in a contiguous region (the array marked **[B]**). Next, a GPU thread is generated for each sorted array element to check whether the assigned photon is the first in the same photon group based on a comparison with the preceding photon in the array, before marking the result in an additional array, which is indicated by **[C]** in the figure. This binary array and the array produced by an exclusive scan of the array (marked **[D]**) together generate the offset information (marked **[E]**), which allow the locations of the first photons and the photon numbers of the enumerated nonempty boundary cells to be determined easily. Next, each GPU thread, where one is spawned for each element of the offset array, i.e., one per nonempty boundary cell, computes the radiance values of all six faces by processing all of the photons in the cell without using any atomic operations.

After the radiance field is produced on the R-grid, we iteratively construct averaged down versions until a field with a suitably coarse level of detail is ob-

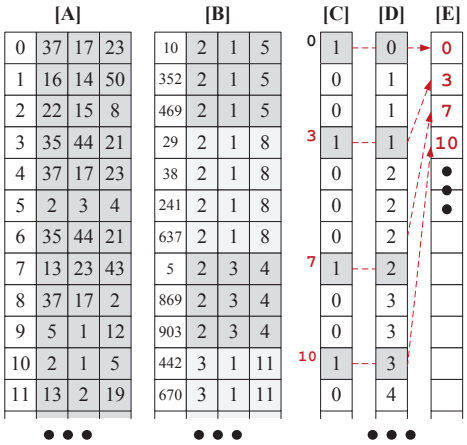


Fig. 3 GPU-assisted photon sorting. A more complicated, but faster, GPU algorithm is used to build the radiance grid, which avoids expensive atomic add operations.

tained. This mipmap-style, multilevel R-grid allows the effective estimation of the reflected radiance in the later rendering stage.

3.4 Construction of a spherical harmonics field

3.4.1 Spherical harmonics for global illumination

Spherical harmonics are an orthogonal basis for functions defined over a sphere, or over a direction, which have proved effective in computer graphics applications, particularly for representing slowly varying indirect illumination. Given a position x in three-dimensional space, we consider an incoming radiance function for indirect diffuse illumination $I(x, \theta, \phi)$ where $\boldsymbol{\omega} = (\theta, \phi)$, or $I(\theta, \phi)$ if the position is evident, which is defined using a spherical coordinate system. For real-valued spherical harmonics $Y_l^m(\theta, \phi)$ of *degree* l and *order* m ($l \in \{0, 1, 2 \cdots\}$, $-l \leq m \leq l$), $I(\theta, \phi)$ can be expanded as their linear combination: $I(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} I_l^m Y_l^m(\theta, \phi)$ with spherical harmonics coefficients I_l^m, which can be evaluated using the projection operation:

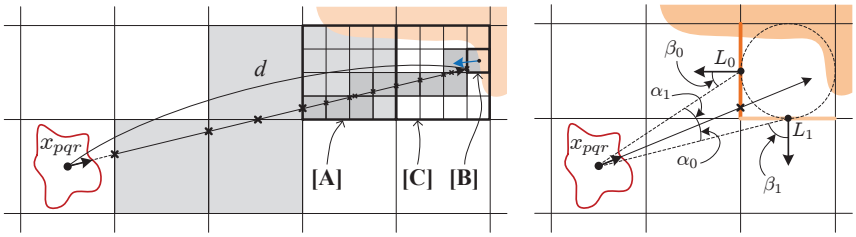
$$I_l^m = \int_0^{2\pi} \int_0^{2\pi} I(\theta, \phi) Y_l^m(\theta, \phi) \sin\theta \, d\theta \, d\phi.$$

In practice, the function $I(\theta, \phi)$ is approximated by $\bar{I}(\theta, \phi)$ using a finite number of coefficients up to a given degree l^*:

$$\bar{I}(\theta, \phi) = \sum_{l=0}^{l^*} \sum_{m=-l}^{l} I_l^m Y_l^m(\theta, \phi).$$

In this case, the $(l^* + 1)^2$ coefficients I_l^m can be approximated by Monte Carlo simulation:

$$I_l^m \approx \frac{4\pi}{N_{sh}} \sum_{i=0}^{N_{sh}-1} I(\theta_i, \phi_i) Y_l^m(\theta_i, \phi_i), \tag{1}$$



(a) Adaptive ray traversal using SH-grid and V-grid (b) Radiance interpolation in R-grid

Fig. 4 Estimation of the incoming radiance for a sample ray. The reflected radiance attributable to diffuse indirect illumination is stored in the directional faces of an R-grid. After a cell in the R-grid that contains an intersection point is identified using the SH-grid and V-grid, the incoming radiance for the ray is interpolated based on the radiance of the visible faces.

where the incoming radiance $I(\theta_i, \phi_i)$ needs to be estimated for N_{sh} uniformly sampled directions (θ_i, ϕ_i).

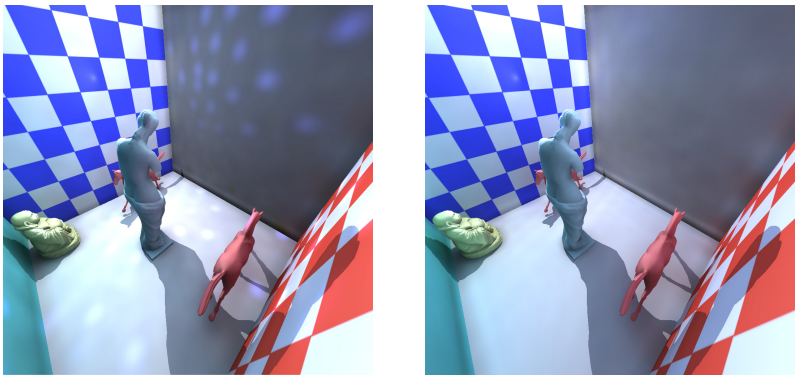
3.4.2 Estimation of incoming radiance at the centers of cells

When the approximate function $\bar{I}(x, \theta, \phi)$ is defined numerically at a surface point x, the expensive final gathering operation around x can be performed efficiently during rendering simply by evaluating the function at stochastically sampled gather ray directions. In contrast to previous methods, e.g., [32,30], where the incoming radiance function is cached at surface points that are selected dynamically during rendering, our method represents the incoming light discretely at the centers of the boundary cells and their 26 neighbors in the SH-grid, which facilitates simpler and faster computation on the GPU.

When the spherical harmonics coefficients I_l^m are evaluated at the center x_{pqr} of a cell with index (p, q, r), the efficient estimation of $I(x_{pqr}, \theta_i, \phi_i)$ for each sample direction is critical for the rendering performance (see Eq. (1)). For effective computation, we employ a simple two-level grid-based ray traversal algorithm. A sample ray from x_{pqr} (see Figure 4(a)) is first traced in the SH-grid space until a boundary cell is hit, other than that containing x_{pqr} (the cell marked **[A]**). Next, the finer V-grid is used to determine more precise ray–object intersections, where the ray continues until a boundary cell (the cell marked **[B]**) is found in the V-grid space. If the identified cell's averaged normal is front-facing, the surface region of the cell is regarded as visible from x_{pqr} and $I(x_{pqr}, \theta_i, \phi_i)$ is interpolated using the radiance values L_j of up to three visible faces of the R-grid cell that contains the hit point (the cell marked **[C]**) as follows:

$$I(x_{pqr}, \theta_i, \phi_i) \approx \sum_j \frac{g(\alpha_j)}{\sum_k g(\alpha_k)} L_j \cos \beta_j,$$

where α_j is the angle between the sample direction (θ_i, ϕ_i) and the direction to the center of the jth face, β_j is the angle between the corresponding face normal and the direction to x_{pqr} from the center, and $g(\cdot)$ is a Gaussian filter function defined on angle α_j (refer to Figure 4(b)).



(a) Without radiance filtering (b) With radiance filtering

Fig. 5 Radiance filtering of incoming indirect light.

3.4.3 Radiance filtering using the multilevel R-grids

Assume that the distance from x_{pqr} to the intersection, which is estimated during ray traversal, is d, as shown in Figure 4(a). Then, each sample ray represents an approximate area of $\frac{4\pi d^2}{N_{sh} \cos \beta_j}$ along the jth face of the intersected R-grid cell. If this area is greater than that of a zero-level R-grid face and the reflected radiance changes rapidly in the vicinity of the sampled R-grid face, there may be aliasing artifacts during the estimation of the incoming radiance $I(x_{pqr}, \theta_j, \phi_j)$ (recall the situation where the mipmapping technique was applied to texture filtering). In our method for radiance estimation, we select two consecutive levels of the multilevel R-grid where the face areas bound the estimated area and we interpolate $I(x_{pqr}, \theta_j, \phi_j)$ for the area using the radiance values extracted from the two levels. Figure 5 compares the rendering results produced without and with radiance filtering, where the noisy artifacts on the wall caused by inappropriate radiance sampling from an overly detailed R-grid are smoothed out well.

3.4.4 Spherical harmonics for distance function estimation

As a by-product of the adaptive grid traversal algorithm, we also obtain an approximate distance $d(x_{pqr}, \theta_j, \phi_j)$ from x_{pqr} to the first hit in the sample direction (θ_j, ϕ_j). This allows us to build a numerical distance function $\bar{d}(x_{pqr}, \theta, \phi)$, which is also based on the spherical harmonics. Note that the scalar coefficient d_l^m forms a four-component vector with the RGB-valued I_l^m, which is well suited to GPU computation. The distance information produced in this stage is then used to create an ambient occlusion-like effect during the rendering stage, which incurs no extra cost because the four-component coefficients are hard-coded in our renderer.

4 Rendering algorithm: the ray tracing stage

4.1 Estimation of incoming radiance at surface points

The method used to construct a spherical harmonics field represents indirect light in the scene, where the numerical functions $\bar{I}(x_{pqr}, \theta, \phi)$ are built at the centers x_{pqr} of the SH-grid cells in the neighborhood of surfaces. During rendering, the radiance function needs to be evaluated at an arbitrary surface point x. To achieve this effectively, we construct the function $\bar{I}(x, \theta, \phi)$ on the fly by computing its spherical harmonics coefficients $I_l^m(x)$ via their interpolation from the approximate radiance functions near x:

$$I_l^m(x) = \sum_{pqr} w_{sh}(d_{pqr}) I_l^m(x_{pqr}) / \sum_{pqr} w_{sh}(d_{pqr}),$$

where $I_l^m(x_{pqr})$ is the corresponding coefficient at the grid center x_{pqr} of 27 neighboring cells around x, d_{pqr} if the distance from x to x_{pqr}, and $w_{sh}(d)$ is a weight function that is defined based on the distance (we find that a simple weight function $w_{sh}(d) = (2/r_{max}^3)d^3 - (3/r_{max}^2)d^2 + 1$ with an appropriate maximum extent r_{max} works well). Then the incoming light at x that is attributable to diffuse indirect illumination can be estimated efficiently by evaluating $\bar{I}(x, \theta, \phi)$ for each ray direction sampled over the hemisphere around x. Figure 6 shows an example.

4.2 Diffuse reflection caused by diffuse indirect illumination

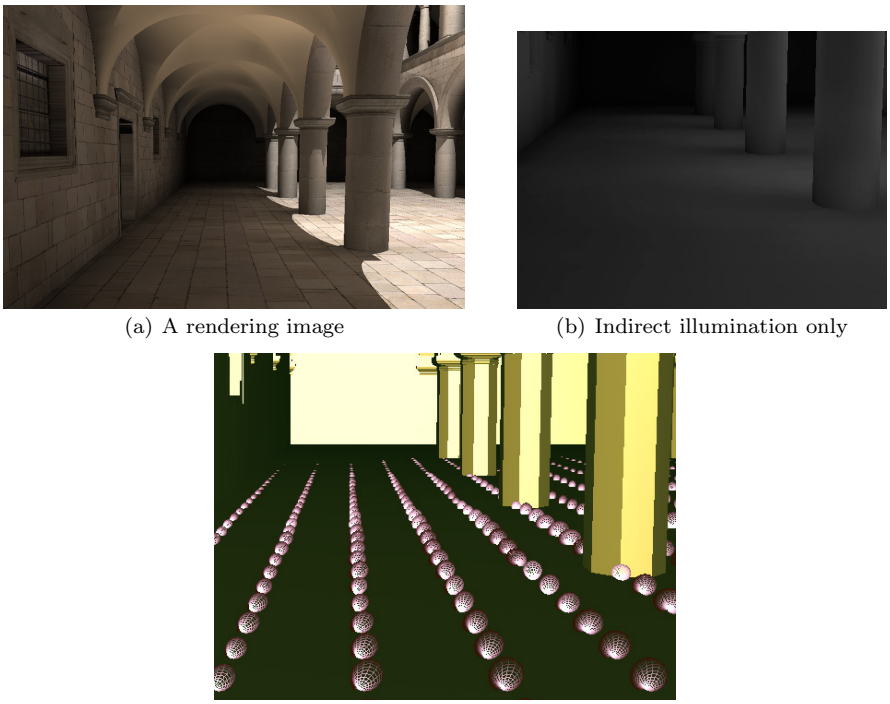
After the preparation stage is complete, the actual rendering process starts by running a Whitted-style ray tracer, which is extended slightly to include the reflection effect attributable to diffuse indirect illumination. We consider a ray–object intersection point x on a diffuse surface with BRDF $f_{r,d}(x)$. The diffuse reflection $L_{ind,d}(x)$ is computed by gathering N_d incoming radiances $I(x, \boldsymbol{\omega})$, which are sampled over the hemisphere around x with a probability density function (PDF) $P(\boldsymbol{\omega})$, and approximated using the spherical harmonics field:

$$L_{ind,d}(x) \approx \frac{1}{N_d} \sum_{i=0}^{N_d-1} \frac{f_{r,d}(x) \bar{I}(x, \boldsymbol{\omega}_i) \cos\theta_i}{P(\boldsymbol{\omega}_i)}.$$

When uniform sampling is applied with $P(\omega) = \frac{1}{2\pi}$, the formula becomes a simple summation:

$$L_{ind,d}(x) \approx \frac{2\pi f_{r,d}(x)}{N_d} \sum_{i=0}^{N_d-1} \cos\theta_i \Big\{ \sum_{l=0}^{l^*} \sum_{m=-l}^{l} I_l^m(x) Y_l^m(\boldsymbol{\omega}_i) \Big\}.$$

To ensure the efficiency of calculation in our implementation, we generate $2N_d$ directions initially by sampling the unit sphere uniformly. During rendering, half of those that point upwards with respect to x are then used to approximate the formula.



(a) A rendering image (b) Indirect illumination only

(c) The constructed SH field

Fig. 6 Reconstruction of diffuse indirect illumination using spherical harmonics. For the Sponza scene in (a), a spherical harmonics field was built to represent indirect light as given in (c), where the RGB-valued radiance $\bar{I}(x_{pqr}, \theta, \phi)$ was converted to grayscale, which is shown only at the cell centers located immediately above the ground. As shown in (b), the shadows around the pillars caused by indirect light were generated naturally by the spherical harmonics field.

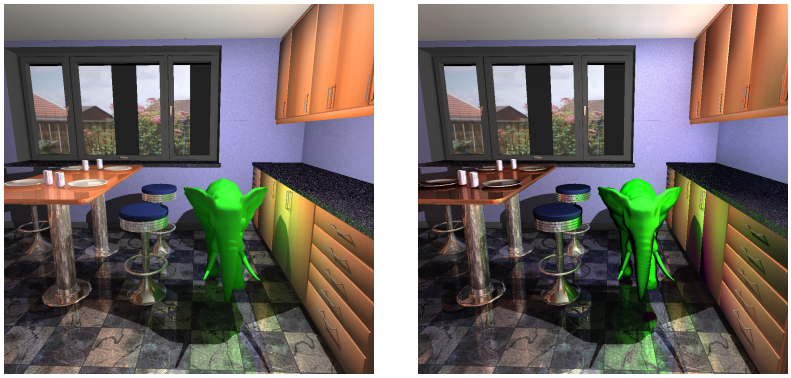
4.3 Glossy reflection caused by diffuse indirect illumination

The Monte Carlo technique can also be applied to approximate a glossy reflection $L_{ind,g}(x, \boldsymbol{\omega})$ caused by diffuse indirect illumination. Given a BRDF $f_{r,g}(x, \boldsymbol{\omega}_i, \boldsymbol{\omega}) = k_s \frac{e+1}{2\pi}(\mathbf{r}_i \cdot \boldsymbol{\omega}_i)^e$ that simulates a glossy reflection, N_g directions can be importance-sampled around the specular reflection direction $\mathbf{r}_i$ using the PDF $P(\boldsymbol{\omega}_i) = \frac{e+1}{2\pi}(\mathbf{r}_i \cdot \boldsymbol{\omega}_i)^e$, through which we obtain a similar formula $L_{ind,g}(x, \boldsymbol{\omega}) \approx \frac{k_s}{N_g} \sum_{i=0}^{N_g-1} \bar{I}(x, \boldsymbol{\omega}_i) \cos\theta_i$ (see Figure 7).

4.4 Creation of an ambient occlusion-like effect

Although it is not physically correct, ambient occlusion provides an interesting rendering effect that adds shadowing to diffuse objects [36]. The approximate distance function allows us to estimate the obscurance $O(x)$ of surface point x, which is defined and estimated as follows: $O(x) = \frac{1}{\pi} \int_{2\pi} \rho(d(x, \boldsymbol{\omega}')) \cos\theta' \, d\boldsymbol{\omega}' \approx \frac{2}{N_d} \sum_{i=0}^{N_d-1} \rho(\bar{d}(x, \boldsymbol{\omega}_i)) \cos\theta_i$, where a control parameter w_{oc} is introduced into

(a) On diffuse surfaces (178.1) (b) On glossier surfaces (188.9)



(c) On diffuse surfaces (212.7) (d) On glossier surfaces (244.1)

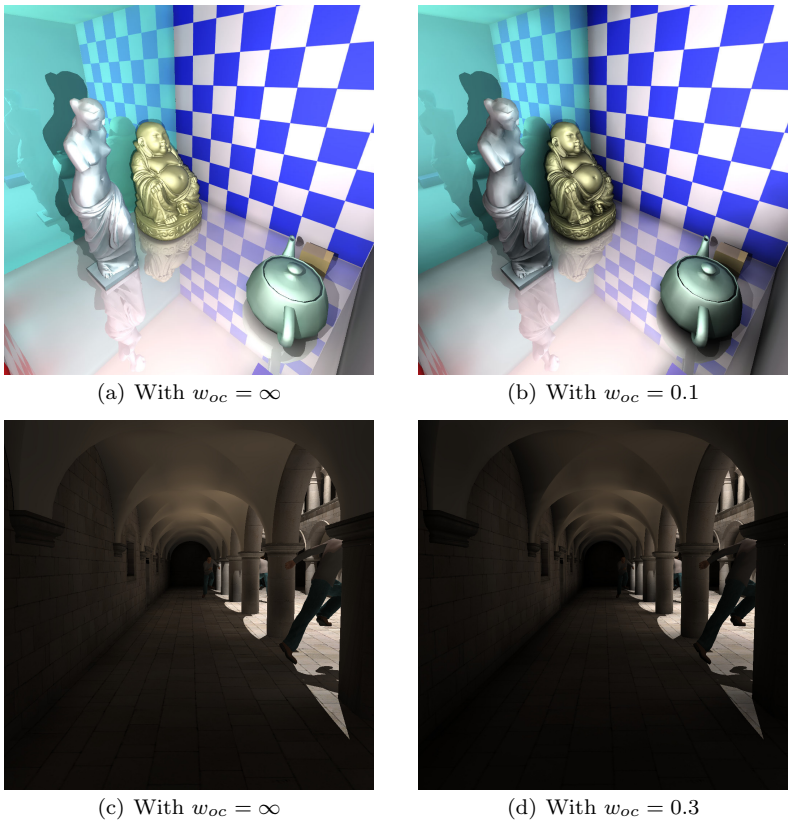
Fig. 7 Interactive reproduction of reflection effects caused by indirect diffuse illumination. In (a), all of the objects in the CBOX scene (76,571 triangles) were set as Lambertian, whereas all but the checker-textured walls were made glossier in (b), where the glossy effect can be observed clearly, particularly on the surface of the statue of Venus. In the kitchen scene (181,141 triangles) in (c) and (d), the reflectances of the elephant, drawers, table, and objects on the table were made glossier. The figures in parentheses show the rendering time in milliseconds for obtaining a 1024×1024 image on an NVIDIA GeForce GTX TITAN GPU.

the definition of $\rho(d) = \max\{ \frac{w_{oc} \cdot d}{R_{max}}, 1 \}$ for a given maximum distance R_{max}, while $\bar{d}(x, \boldsymbol{\omega}_i)$ is again estimated using the precomputed spherical harmonics field. The approximated obscurance value is then multiplied by $L_{ind,d}(x)$ to obtain the wanted effect (see Figure 8).

5 Experimental results

5.1 Rendering cost

To simulate the proposed method and demonstrate its effectiveness, we extended a Whitted-style ray tracer in the CUDA platform by including diffuse indirect illumination and tested its performance using several dynamic scenes with non-trivial complexity. Table 1 summarizes the statistics collected on a desktop PC



(a) With $w_{oc} = \infty$ (b) With $w_{oc} = 0.1$

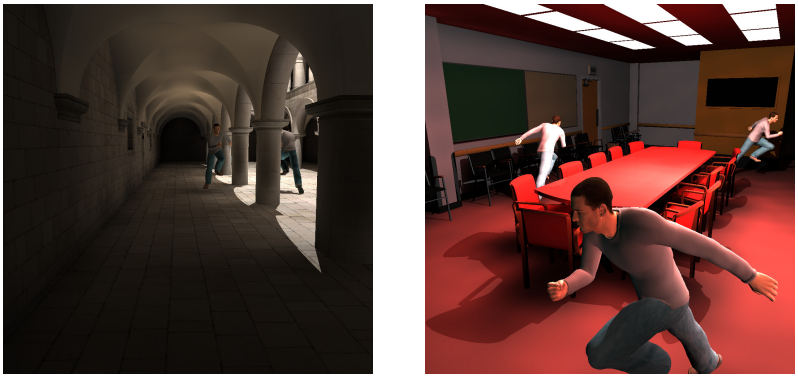
(c) With $w_{oc} = \infty$ (d) With $w_{oc} = 0.3$

Fig. 8 Easily controllable ambient occlusion-like effect. We can generate a visually pleasing rendering effect by using an extra set of spherical harmonics coefficients to approximate the distance function.

with an NVIDIA GeForce GTX TITAN GPU for the four example scenes shown in Figure 9, where the timings were measured while ray tracing a $1,024 \times 1,024$ image using CUDA blocks with 4×32 threads. In this experiment, to ensure efficient ray–object intersections were obtained during ray tracing, we employed a two-tree scheme where a kd-tree was built only once for a static geometry, while a bounding volume hierarchy was updated for each frame for dynamic objects. We also used the voxelization technique proposed by Schwarz and Seidel [21] to build the V-grid interactively.

To simulate indirect light transport, 262,144 photons were emitted initially from the light sources in each test scene, where the conventional Russian roulette technique was applied during photon tracing. However, we processed a maximum of four photon reflections, which actually provided five reflections including the final gathering step, because a higher number of reflections generally produced no visual differences. The incoming radiance $I(x, \theta, \phi)$ was modeled using third-degree spherical harmonics, the nine coefficients of which were estimated by Monte Carlo simulation using 400 direction samples ($N_{sh} = 400$). The diffuse and glossy reflections, i.e., $L_{ind,d}(x)$ and $L_{ind,g}(x, \boldsymbol{\omega})$, caused by indirect light were also sim-

(a) Kitchen (181,141) (b) Bathroom (273,750)



(c) Sponza (300,541) (d) Conference (425,024)

Fig. 9 Rendering results for four test scenes. To experiment with dynamic scenes of nontrivial complexity, we synthesized these scenes from commonly used examples: the kitchen and bathroom scenes were augmented with an elephant and a horse model, respectively. Three copies of Ben models were added to the Sponza and conference room scenes. The figures in parentheses represent the final number of triangles in the scenes.

ulated using 32 and 64 direction samples, respectively ($N_d = 32$ and $N_g = 64$). When combined with the spherical harmonics tool, this small number of stochastic samples was sufficient to generate the global illumination effect, which compared favorably with that obtained using the computationally intensive final gathering method that often requires several thousand gather rays. For the parameter r_{max} in the weight function $w_{sh}(d)$, we observed that a SH-grid interval of 1.75 times usually worked well.

Overall, our extended ray tracer achieved a frame rate of 3 to 6 frames per second when rendering the example scenes, and it was particularly successful at reproducing intractable diffuse indirect illumination. Note that the interactive global illumination method [30] rendered images at 1.5 frames per second on the old NVIDIA GeForce GTX 280 GPU (933.12 GFLOPS) for a rather small Kitchen scene with 21K triangles. While it is not easy to directly compare the method to ours, we believe that a significant improvement has been made by the presented method considering that ours achieves over 5 frames per second on the tested

Fig. 10 Reflection caused by diffuse indirect illumination.

GPU (4,500 GFLOPS (single precision) and 1,500 GFLOPS (double precision)) for the scenes with over 10 times more triangles and produces higher-quality rendering effects.

On the other hand, the step-by-step analysis of the rendering times shown in Table 1 demonstrates that the rendering performance depended on several parameters. First, the voxelization cost in the *Voxelization* row of the table was affected greatly by the relative sizes of triangles and their variances relative to the V-grid cell size because one GPU thread was wholly responsible for voxelizing an assigned triangle in the algorithm we used [21]. The voxelization performance was relatively poor for the Sponza scene, which contained very large (walls) and small (running men) triangles, and this suggests that the use of well-tessellated triangles may give greater efficiency. Second, the photon tracing time in the *Photon tracing* row generally increased as more photons were stored on surfaces. However, the behavior was rather complex because the parallel traversal of the spatial acceleration structure on the GPU when tracing incoherent photon rays was also affected by the scene geometry.

Third, as expected, the final two steps of the preparation stage used to convert the diffuse indirect illumination into a computationally efficient form required a significant amount of computational time, as shown in the two rows for *multilevel R-grid construction and spherical harmonics field construction*. Interestingly, the lighting condition and the GPU computing architecture were reflected in the parallel performance of R-grid construction. For example, in the dark room in the Conference scene, the global photons were sorted highly heterogeneously in the R-grid cells (see Figure 10 for the rendering of indirect illumination only), so the few parallel threads assigned to the highly lit R-grid cells incurred high overheads, which created a bottleneck that degraded the GPU performance.

Finally, as shown in the *extended ray tracing* row, the burden of the ray tracing stage varied according to the scene geometry and the rendering parameters, in the same way as conventional ray tracing. Our ray tracer performs basic Whitted-style ray tracing, but with the addition of a global illumination computation using the spherical harmonics field. The ray tracing times required for pure Whitted-style ray tracing calculations are given in the *[Pure Whitted-style ray tracing]* row, which indicates that the additional cost of interpolating the spherical harmonics

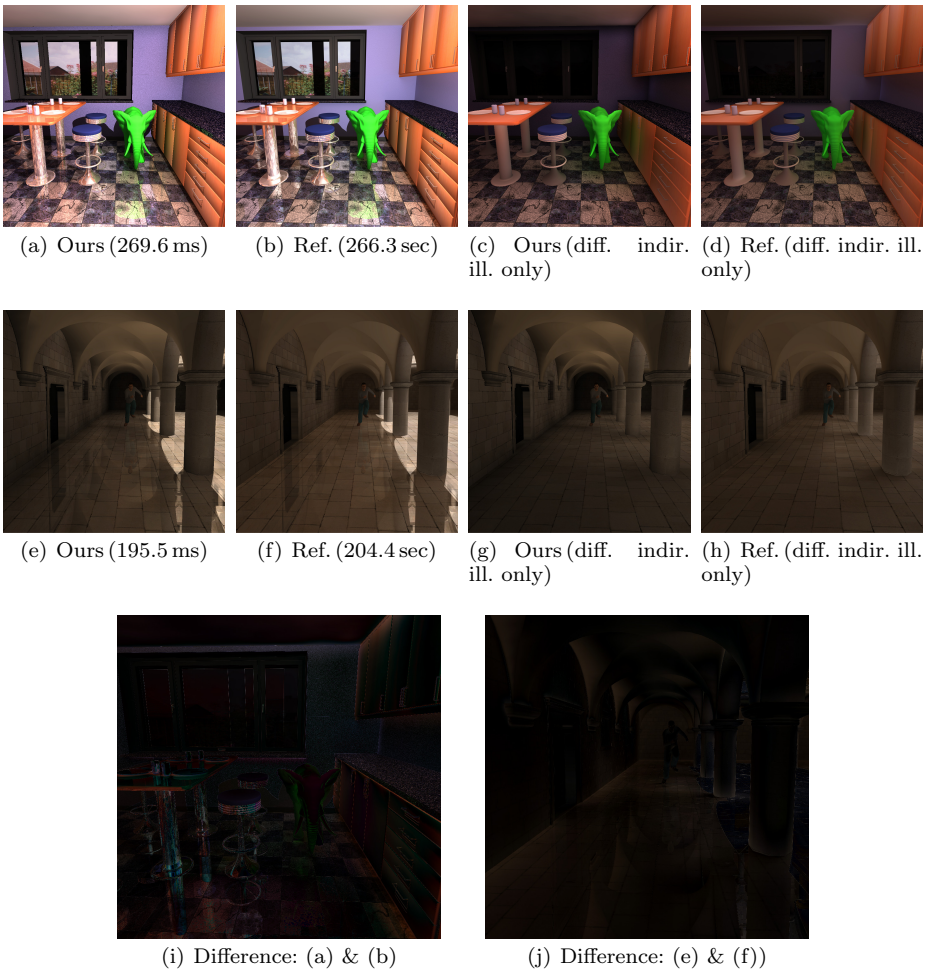
	Kitchen	Bathroom	Sponza	Conference
# of triangles	181,141	273,750	300,541	425,024
SH-grid resolution	32^3	32^3	64^3	64^3
V-grid resolution	128^3	128^3	256^3	256^3
Total rend. time (ms / fps)	212.7 / 4.70	207.3 / 4.82	182.8 / 5.47	339.3 / 2.95
Voxelization [# of boundary cells]	1.8% [5,960]	4.5% [13,197]	14.0% [26,479]	3.5% [26,543]
BVH update	0.1%	0.1%	0.4%	0.3%
Photon tracing [# of stored photons]	8.1% [528,936]	9.0% [657,580]	6.1% [259,550]	2.4% [325,082]
Multi-level R-grid const.	31.0%	34.5%	13.8%	56.0%
SH field construction	13.1%	23.0%	45.5%	23.4%
Extended ray tracing [Pure Whitted-style]	45.9% [76.9%]	28.9% [61.9%]	20.2% [68.6%]	14.4% [73.8%]

Table 1 Rendering statistics. The total rendering time is broken down into the separate steps and their relative overheads are provided for in-depth analysis.

coefficients and gathering the incoming radiances comprised 23.1% to 38.1% of the entire ray tracing computation, which was affected mainly by the amount of indirect illumination in the scenes.

5.2 Image quality

The main focus of our proposed method was the effective depiction of the reflection of incoming indirect light, which is difficult to trace, particularly because of light-scattering diffuse interreflection. To compare our results with reference images produced by an offline renderer, we produced a full implementation of a GPU-assisted path tracer where the photon mapping and final gathering techniques were used for efficient rendering. We used 4,096 gather rays per shading point to create the final ground truth images (see Figure 11). A thorough analysis showed that our method produced satisfactory global illumination results, which compared favorably with the reference images, and it was often difficult to detect visual differences when they were rendered interactively. In particular, the interpolation of spherical harmonics in the three-dimensional space around the shading points produced very few temporal aliases during dynamic scene changes, although we only used 32 gather rays. Like all numerical discretization techniques, however, our rendering algorithm had approximation errors because of spatial discretization and the low-order spherical harmonics. The most obvious visual error usually occurred around the corners or highly curved objects, where the applied grid resolution was not sufficient to discretize the rapid geometry and light changes faithfully (refer to Figure 11 (i) and (j) to see the difference between our result and the reference image). A possible solution to this problem may be the use of a higher resolution grid, although this would increase the rendering cost, which strongly suggests that it would be beneficial to apply the multilevel adaptive grid technique to the grids, in addition to the R-grid.



(a) Ours (269.6 ms) (b) Ref. (266.3 sec) (c) Ours (diff. indir. ill. only) (d) Ref. (diff. indir. ill. only)

(e) Ours (195.5 ms) (f) Ref. (204.4 sec) (g) Ours (diff. indir. ill. only) (h) Ref. (diff. indir. ill. only)

(i) Difference: (a) & (b) (j) Difference: (e) & (f))

Fig. 11 Comparison of our proposed method with reference images produced by an offline GPU-assisted path tracer that supported photon mapping and final gathering. The path tracer used a pixel sampling and shading strategy that was inherently different from our interactive ray tracer. Therefore, there were slight differences in the appearance of the rendering images, although the visual differences were often difficult to detect when rendered interactively.

6 Concluding remarks

The key to the interactive global illumination of fully dynamic scenes with an image quality similar to final gathering is the efficient representation of the incoming radiance caused by indirect light in the vicinity of surfaces and performing the final gathering step with a minimal cost during rendering. Using a multilevel grid-based scheme and the mathematical tool of spherical harmonics, our rendering method extended the Whitted-style ray tracer successfully to handle interactive intractable diffuse indirect illumination. The straightforward computational structure of our system meant that it was easy to implement on the GPU without relying on

complex global illumination techniques such as irradiance caching, which is often vulnerable to temporal flickering artifacts. At present, our rendering algorithm is limited by the memory required to store various volume grids and the spherical harmonics coefficients. Thus, we are developing an effective multilevel algorithm to represent the V-grid and SH-grid adaptively, which will increase the effective grid resolution, as well as using higher-order spherical harmonics to further enhance the rendering quality without degrading the level of interactivity.

Acknowledgements The test scenes are courtesy of J. Helenklaken (Kitchen), M. Dabrovic (Sponza), A. Grynberg and G. Ward (Conference), I. Wald (Ben), and R. Sumner and J. Popovic (Elephant and Horse). 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 T GRAPHIC*, 24(3):1108–1114, 2005.
2. Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. *COMPUT GRAPH FORUM*, 30(7):1921–1930, 2011.
3. C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, pages 203–231, 2005.
4. B. Fabianowski and J. Dingliana. Interactive global photon mapping. *COMPUT GRAPH FORUM*, 28(4):1151–1159, 2009.
5. P. Gautron, J. Křivánek, K. Bouatouch, and S. Pattanaik. Radiance cache splatting: a GPU-friendly global illumination algorithm. In *Proc. Eurographics Symp. on Rendering*, pages 55–64, 2005.
6. R. Green. Spherical harmonic lighting: The gritty details. In *Proc. GDC*, 2003.
7. Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proc. 18th Eurographics Conf. on Rendering Techniques*, pages 35–44, 2007.
8. H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Ltd., 2001.
9. Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, pages 99–107, 2010.
10. Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed local radiance transfer for real-time lighting design. *ACM T GRAPHIC*, 24(3):1208–1215, 2005.
11. J. Křivánek, P. Gautron, S. Pattanaik, and K. Bouatouch. Radiance caching for efficient global illumination computation. *IEEE T VIS COMPUT GR*, 11(5):550–561, 2005.
12. B. Larsen and N. Christensen. Simulating photon mapping for real-time applications. In *Proc. Eurographics Symp. on Rendering*, pages 123–131, 2004.
13. David Maletz and Rui Wang. Importance point projection for GPU-based final gathering. In *Proc. of Eurographics Symp. on Rendering*, pages 1327–1336, 2011.
14. Pavlos Mavridis and Georgios Papaioannou. Global illumination using imperfect volumes. In *Proc. GRAPP*, pages 160–165, 2011.
15. M. McGuire and David D. Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proc. High Performance Graphics*, pages 77–89, 2009.
16. Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Real-time global illumination on the GPU. *J GRAPHIC TOOL*, 10:55–71, 2004.
17. Georgios Papaioannou. Real-time diffuse global illumination using radiance hints. In *Proc. High Performance Graphics*, pages 15–24, 2011.
18. T. Purcell, C. Donner, M. Cammarano, H. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proc. Graphics Hardware*, pages 41–50, 2003.
19. T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. The state of the art in interactive global illumination. *COMPUT GRAPH FORUM*, 31(1):160–188, 2012.

20. T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM T GRAPHIC*, 28(5), 2009. Article No. 132.
21. M. Schwarz and H.-P. Seidel. Fast parallel surface and solid voxelization on GPUs. *ACM T GRAPHIC*, 29:179:1–179:9, 2010.
22. P.-P. Sloan. Stupid spherical harmonics tricks, 2008.
23. Peter-Pike Sloan, Naga K. Govindaraju, Derek Nowrouzezahrai, and John Snyder. Image-based proxy accumulation for real-time soft global illumination. In *Proc. Pacific Graphics '07*, pages 97–105, 2007.
24. Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM T GRAPHIC*, 22(3):382–391, 2003.
25. Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM T GRAPHIC*, 21(3):527–536, 2002.
26. E. Tabellion and A. Lamorlette. An approximate global illumination system for computer generated films. *ACM T GRAPHIC*, 23(3):469–476, 2004.
27. Sinje Thiedemann, Niklas Henrich, Thorsten Grosch, and Stefan Müller. Voxel-based global illumination. In *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, pages 103–110, 2011.
28. T. Umenhoffer and L. Szirmay-Kalos. Robust diffuse final gathering on the GPU. In *Proc. WSCG*, 2007.
29. Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM T GRAPHIC*, 24(3):1098–1107, July 2005.
30. R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao. An efficient GPU-based approach for interactive global illumination. *ACM T GRAPHIC*, 28, 2009. Article No. 91.
31. G. Ward and P. Heckbert. Irradiance gradients. In *Proc. Eurographics Workshop on Rendering*, pages 85–98, 1992.
32. G. Ward, F. Rubinstein, and R. Clear. A ray tracing solution for diffuse interreflection. In *Proc. ACM SIGGRAPH*, pages 85–92, 1988.
33. T. Whitted. An improved illumination model for shaded display. *COMMUN ACM*, 23:343–349, 1980.
34. C. Yao, B. Wang, B. Chan, J. Yong, and J.-C. Paul. Multi-image based photon tracing for interactive global illumination of dynamic scenes. *COMPUT GRAPH FORUM*, 29:1315–1324, 2010.
35. K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time KD-tree construction on graphics hardware. *ACM T GRAPHIC*, 27(5):1–11, 2008.
36. S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In *Proc. Rendering Techniques '98*, pages 45–55. 1998.