Global Illumination for Interactive Lighting Design using Light Path Pre-computation and Hierarchical Histogram Estimation

Yonghao Yue
The University of Tokyo
yonghao@nis-lab.is.s.u-tokyo.ac.jp

Kei Iwasaki
Wakayama University
iwasaki@sys.wakayama-u.ac.jp

Yoshinori Dobashi
Hokkaido University
doba@nis-ei.eng.hokudai.ac.jp

Tomoyuki Nishita
The University of Tokyo
nis@nis-lab.is.s.u-tokyo.ac.jp

Abstract

In this paper, we propose a fast global illumination solution for interactive lighting design. Using our method, light sources and the viewpoint are movable, and the characteristics of materials can be modified (assuming low-frequency BRDF) during rendering. Our solution is based on particle tracing (a variation of photon mapping) and final gathering. We assume that objects in the input scene are static, and pre-compute potential light paths for particle tracing and final gathering. To perform final gathering fast, we propose an efficient technique called Hierarchical Histogram Estimation for rapid estimation of radiances from the distribution of the particles. The rendering process of our method can be fully implemented on the GPU and our method achieves interactive frame rates for rendering scenes with even more than 100,000 triangles.

1. Introduction

Today, various applications, such as movies, lighting design, games and virtual reality, make use of the calculation of the global illumination for the purpose of improving the realism in rendered images by capturing lighting effects such as the inter-reflections of light. One of the challenging goals in Computer Graphics is to compute global illumination at interactive frame rates.

In this paper, we propose a fast global illumination solution for lighting design, where it is reasonable to assume that the objects in the input scene are static. This means that potential light paths can be pre-computed, with the energies they carry being recalculated only when the viewpoint or light sources move. Our system deals with diffuse and low frequency glossy BRDFs (Bi-directional Reflectance Distribution Functions), and can handle various types of light sources. The features of our method are as follows:

- light sources and the viewpoint can be moved at interactive frame rates,
- materials (including the textures and BRDFs) of objects in the scene can be changed at run-time.

We use particle tracing and final gathering to compute global illumination. We pre-compute potential light paths, and propose an effective algorithm called hierarchical histogram estimation for fast final gathering. Using our method, there is no need to perform ray tracing during rendering, and the radiance estimation via particles can be performed with only a few array accesses. Moreover, the rendering process can be fully implemented on the GPU.

The rest of this paper is organized as follows. Section 2 reviews related previous methods. Sections 3 and 4 describe the system flowchart and the theoretical background of our method, respectively. The detailed process of the pre-computation and rendering are described in Sections 5 and 6, respectively. In Section 7, we describe our GPU implementation. We show the results and limitations in Sections 8 and 9, respectively. The conclusions and the future work are brought together in Section 10.

2. Previous Work

Global illumination solutions: The goal of the global illumination is to solve the rendering equation proposed by Kajiya [12]. Some well-known approaches are the Radiosity solutions [17, 4, 7, 5] and the Monte Carlo based solutions [26]. We will not focus on Radiosity solutions since they often have difficulties in dealing with specular (glossy) surfaces. To improve the rendering efficiency in Monte Carlo based solutions, multi-pass methods were introduced. One of the most famous multi-pass methods is photon mapping [11]. Other methods that accelerate the rendering are the irradiance caching methods [27, 23] and the radiance caching method [14]. These methods exploit the coherence in the image space, and use interpolation schemes to reduce expensive radiance evaluations.

Final gathering: This method estimates the radiance distribution incident onto a calculation point by shooting many sampling rays from the calculation point. Tawara et al. [25] introduced an algorithm that exploits coherence in the temporal domain to reduce the number of sampling rays. Scheel et al. [20] proposed a grid based final gathering that stores irradiance in hierarchical grids to accelerate the rendering of diffuse surfaces. Christensen et al. [3] proposed to store irradiance in hierarchical grids to accelerate the rendering of diffuse surfaces. Arikan et al. [1] accelerated final gathering by decomposing the radiance field into far and near field components. These methods, however, did not achieve interactive rendering.

Path re-using: Selective photon tracing [6] uses coherence between frames to re-scatter photons only into some critical regions where the illumination distributions change significantly. Bekaert et al. [2] presented an acceleration path tracing algorithm by re-using paths for the reduction of noise. Sbert et al. [19] re-used light paths to generate animations of moving light sources in real-time by using graphics hardware. A limitation in this method that the viewpoint is fixed was removed by Szécsi et al. [22]. These methods, however, can deal with diffuse surfaces only and cannot change surface materials at run-time.

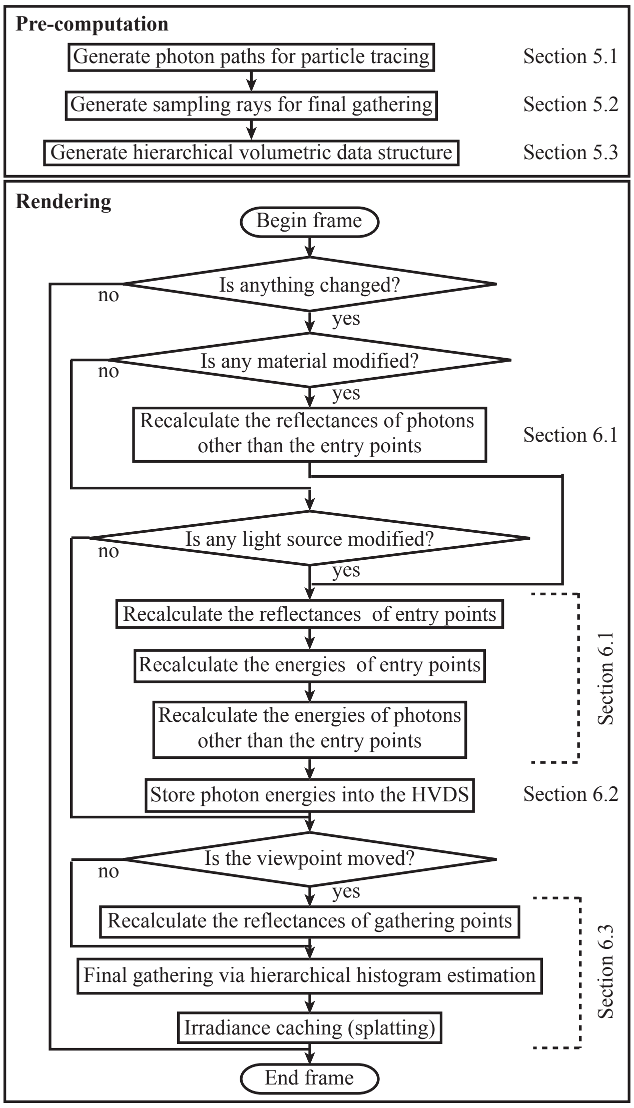
Real-time re-lighting using PRT: Pre-computed Radiance Transfer (PRT) was first developed by Sloan et al. [21] to re-light a scene in real-time, and a lot of variants were proposed, including using wavelet bases to handle all-frequency light transport [16]. Kristensen et al. [13] proposed a PRT method for local light sources. Their method needs a huge amount of memory for the pre-computed data, and can handle only isotropic point light sources. Hašan et al. [8] used wavelet formulation for final gathering, and achieved high-quality rendering at interactive frame rates under the assumption of a fixed viewpoint and fixed material characteristics. Iwasaki et al. [10] extended shadow fields methods [28, 24], and achieved real-time rendering with global illumination even when moving objects rigidly or modifying materials. The rendering frame rates of their method, however, drop rapidly as the number of objects increases.

3. System Flowchart

In this section, we show the system flowchart of our method, and indicate the corresponding section that describes the details of each process. Some definitions of important terms used in the rest of this paper are shown in Figure 1, and the system flowchart is shown in Figure 2. An intuitive illustration about our particle tracing is shown in Figure 3. We would like the readers to notice that the definition of the term *photon path* is different from some previous literature, and is not to be confused.

Term definitions	
Photons:	particles in the particle tracing.
Photon paths:	the paths of the particles. The vertices belonging to light sources are **NOT** included.
Entry points:	a special class of photons, the first vertices of photon paths.
Gathering rays:	sampling rays for final gathering.
Gathering points:	locations where we shoot gathering rays.
Hierarchical volumetric data structure (HVDS):	hierarchical 3D voxels to store radiance distribution.
Grid-cell:	a 3D voxel of the HVDS.
Hierarchical histogram estimation:	our radiance estimation technique using the HVDS.

Figure 1. Term definitions.



Pre-computation
Generate photon paths for particle tracing
Section 5.1
Generate sampling rays for final gathering
Section 5.2
Generate hierarchical volumetric data structure
Section 5.3
Rendering
Begin frame
Is anything changed?
no
yes
Is any material modified?
no
yes
Recalculate the reflectances of photons other than the entry points
Section 6.1
Is any light source modified?
no
yes
Recalculate the reflectances of entry points
Recalculate the energies of entry points
Recalculate the energies of photons other than the entry points
Section 6.1
Store photon energies into the HVDS
Section 6.2
Is the viewpoint moved?
no
yes
Recalculate the reflectances of gathering points
Final gathering via hierarchical histogram estimation
Irradiance caching (splatting)
Section 6.3
End frame

Figure 2. System flowchart.

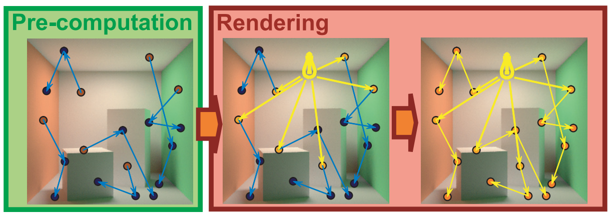


Figure 3. Particles are traced without the consideration of light sources during pre-computation. Energies of particles are recalculated after the locations of light sources are determined during rendering.

4. Theoretical Background

In order to achieve our goal, there are two problems to be solved. The first one is how to pre-compute light paths and use them during rendering even when we have no knowledge about the materials during pre-computation. We review the Monte Carlo integration, and show a way to deal with this problem. The second one is how to efficiently estimate the radiances form the distribution of photon energies during final gathering. Traditional photon mapping uses nearest neighbor searching of photons, which is prohibitively time consuming for the purpose of interactive rendering. We propose hierarchical histogram estimation to solve the second problem.

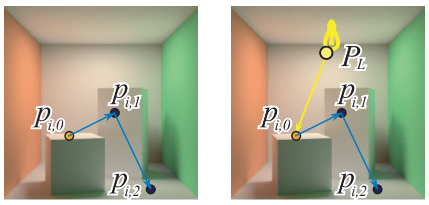
4.1. Contribution of photons

Assuming we are to compute the integration of a function $f(x)$ in a domain D, then the integrand I is given by $I = \int_D f(x)dx$. The Monte Carlo approximation of I is calculated by sampling x at a certain pdf (probability density function) $P(x)$, and calculating the weighted sum by

$$I = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{P(X_i)}, \qquad (1)$$

where X_i is the random variable. For computing global illumination, this corresponds to sample a large number of light paths in order to compute the color of each pixel. Veach [26] formulated the contribution function for a light path. In his case, X_i in Eq.(1) is the random variable in the path domain, and $f(X_i)$ is the contribution funciton. By following his discussion, we can compute the contribution of each photon.

We consider light paths that are sampled without the considerarion of light sources or the viewpoint. Each of these paths connects two arbitrary locations on the surfaces of objects, allowing an arbitrary number of bounces (see Figure 4(a)). These light paths are used in our particle tracing, and



(a) (b)

Figure 4. (a): A photon path. (b): The photon path is connected to a light source.

are called *photon paths*. Assume a photon path p_i is sampled starting from $p_{i,0}$ (the *entry point*), and the remaining vertices are $p_{i,1}$, $p_{i,2}$, $\cdots$, $p_{i,m-1}$, where m is the number of vertices of p_i. Then the contribution $C_{i,k}$ of the photon $p_{i,k}$ in the photon path connected to a vertex P_L on a light source (Figure 4(b)) is given by,

$$C_{i,k} = L_e(P_L \rightarrow p_{i,0})G(P_L \leftrightarrow p_{i,0})\frac{f_s(P_L \rightarrow p_{i,0} \rightarrow p_{i,1})}{P_\sigma^\perp(P_L \rightarrow p_{i,0} \rightarrow p_{i,1})} \cdot \prod_{j=1}^{k-2}\frac{f_s(p_{i,j-1} \rightarrow p_{i,j} \rightarrow p_{i,j+1})}{P_\sigma^\perp(p_{i,j-1} \rightarrow p_{i,j} \rightarrow p_{i,j+1})}, \qquad (2)$$

where L_e is the radiance emitted from P_L toward the entry point, $G(P_L \leftrightarrow p_{i,0})$ is the geometry term between P_L and $p_{i,0}$ (see Section 6.1 for detailed definition), f_s is the BRDF, and $P_\sigma^\perp$ is the pdf used to sample the photon path. We used the notation $P_\sigma^\perp$ rather than P since the cosine term is included implicitly (see [26] for the detail). We would like to mention that the pdf $P_\sigma^\perp$ can be chosen arbitrarily as long as $f_s > 0 \Rightarrow P_\sigma^\perp > 0$ holds. Therefore, photon paths can be pre-computed with certain pdfs, and the contributions can be re-computed correctly during rendering by recalculating the BRDFs even when we change a material. Since we have no knowledge about the BRDFs during pre-computation, we choose the pdf to be the cosine term. Note that if a certain material is chosen to be fixed during rendering, then we can select the pdf according to the BRDF, which will make the computation of glossy surfaces more accurate.

4.2. Hierarchical histogram estimation

Following the rendering equation, the outgoing radiance $L_o(x, \omega_o)$ at the calculation point x towards the direction ω_o reaching the viewpoint is calculated by estimating the radiances incident onto the calculation point. This can be described as follows:

$$L_o(x, \omega_o) = \int_\Omega L_i(x, \omega_i) f_s(x, \omega_i \rightarrow \omega_o) d_{\omega_i}^\perp, \qquad (3)$$

where Ω is the upper hemisphere aligned to the normal at x, $L_i(x, \omega_i)$ is the radiance of light incident from the direction ω_i, $f_s(x, \omega_i \rightarrow \omega_o)$ is the BRDF at x, and $d_{\omega_i}^\perp$ is

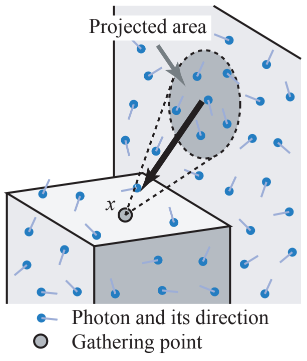


Figure 5. Collecting photon energies.

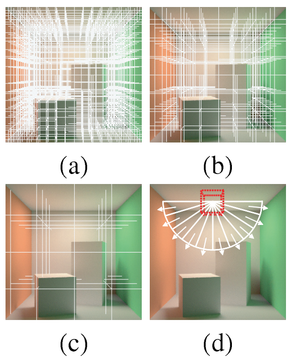


Figure 6. Hierarchical volumetric data structure.

the projected solid angle (with the cosine term implicitly included).

We rewrite Eq.(3) by subdividing Ω into solid angles $\Omega_j (j = 1, ..., N)$ as

$$L_o(x, \omega_o) = \sum_{j=1}^{N} \int_{\Omega_j} L_i(x, \omega_i) f_s(x, \omega_i \to \omega_o) d^{\perp}_{\omega_i}, \quad (4)$$

where N is the number of gathering rays, and Ω_j satisfies $\sum_{j=1}^{N} \Omega_j = \Omega$. We approximately calculate Eq.(4) by assuming the BRDF $f_s(x, \omega_j \to \omega_o)$ to be constant within the solid angle Ω_j (this is simply a Monte Carlo approximation, and L_o will converge to the correct value as we increase N). That is,

$$L_o(x, \omega_o) = \sum_{j=1}^{N} f_s(x, \omega_j \to \omega_o) L_j, \quad (5)$$

where ω_j is the representative direction of the solid angle Ω_j, and

$$L_j = \int_{\Omega_j} L_i(x, \omega) d^{\perp}_{\omega}, \quad (6)$$

represents the total radiance incident from the solid angle Ω_j. To compute L_j, a sampling ray in direction $-\omega_i$ is shot from x, and the hit point y of the sampling ray is calculated. Then, the energies of photons within the projected area of Ω_j at y are collected (see Figure 5).

As shown in Figure 6, a bounding box of the scene is divided into 3D voxels. We collect the photon energies efficiently by accumulating the photon energies into the 3D voxels after recalculating photon energies, and by using the accumulated energies stored in the voxel including y. We call each voxel a *grid-cell*. There are still two things to deal with. The first one is to handle glossy surfaces, and the second one is to take into account the variation in projected areas.

To deal with glossy surfaces, we discretize outgoing directions of photons in world coordinates, and accumulate the photon energies per each discretized outgoing direction (Figure 6(d)).

To deal with the second problem, we use an octree-like hierarchical representation of the voxels, in other words, the hierarchical volumetric data structure (HVDS). An appropriate grid-cell is selected for a projected area. We calculate the projected area by using ray differentials [9]. We show an example of the HVDS in Figures 6 (a) to (c). Note that in this example, the base level (the finest level) is created by uniform partitioning, whereas we will propose an adaptive subdivision construction technique in Section 5.2.

We call the above processes for radiance estimation, Hierarchical Histogram Estimation, since the accumulated energy distribution is intuitively like a histogram, and the hierarchical representation is used for radiance estimation.

Now, we can describe the processes of pre-computation and rendering (described in Sections 5 and 6, respectively), having the theoretical background described above.

5. Pre-computation

In this section, we describe the pre-computation process for photon paths, gathering rays, and the hierarchical volumetric data structure.

5.1. Pre-computation of photon paths

To pre-compute the photon paths, we first determine the locations of entry points. In lighting design, the user may frequently try the arrangement of light sources in a limited region, for example, ceilings. To calculate accurately the intensity distributions in such a region, our method determines the locations of entry points adaptively. Initially, entry points are distributed uniformly on the surfaces of the scene. To determine where to add entry points, our method considers two conditions, 1) the number of visible entry points seen from a light source exceeds a threshold, and 2) the distribution of the entry points seen from a light source should be as uniform as possible. To assure these two conditions, we uniformly subdivide the user's intended region into grids, and check the number and the distribution of visible entry points seen from each grid point. If the distribution is highly skewed or the number is not sufficient, we then adaptively add new entry points.

After the entry points are distributed, we check the nearest neighbors of each entry point and pre-compute the differential area for each entry point, since we use the area to calculate photon energy during rendering. Note that if the entry points are sampled uniformly, the differential area is simply the total area of the triangles in the scene divided by the number of the entry points.

Then, we start particle tracing from each entry point. The outgoing directions of entry points are selected randomly according to the cosine term. For the other photons, the directions are chosen according to particular BRDFs if the corresponding materials are assumed to be fixed during rendering, or to the cosine term, otherwise. We pre-compute the photon paths with length up to a particular number, in this paper, which is 3.

Since the energies of the photons are stored in grid-cells of the HVDS, the index of the grid-cell of the finest level and the index of the discretized outgoing direction are stored for each photon. We store the reflectance as well as the indices for fast recalculation of photon energies during rendering, since the reflectances will remain the same unless the user modifies the corresponding materials. We assign 5 bytes to the index of the grid-cell, 2 bytes to the index of the outgoing direction, and 3 bytes to the reflectance. To efficiently recalculate the reflectances of entry points as the user moves a light source, the geometric data consists of the index of the triangle and two barycentric coordinates are also stotred for each entry point.

5.2. Pre-computation of gathering rays

To pre-compute gathering rays, we first determine the locations of gathering points. We use a rejection-sampling technique to find appropriate locations. That is, we densely sample candidate points for the gathering points, and then discard those where the radiance can likely be interpolated from the radiances at other points. We use a modified error metric proposed by Tabellion et al. [23] to do this. The error metric $E_i(x,n)$ estimates the error when using the radiance at a gathering point x_i to interpolate the radiance at a candidate point x with normal n, and is given by

$$\begin{aligned} E_i(x,n) &= \kappa \cdot max(E_{pi}(x), E_{ni}(x)), \quad (7)\\ E_{pi}(x) &= \frac{|x-x_i|}{max(min(R_i/2, R_+), R_-)},\\ E_{ni}(n) &= \frac{\sqrt{1-n\cdot n_i}}{\sqrt{1-\cos(\alpha_+)}}, \end{aligned}$$

where n_i is the normal at x_i. x is added as a new gathering point if, and only if $E_i(x,n) \geq 1$. The parameters except for R_+ and R_- are equal to those proposed in [23]. That is, $\kappa = 1$, $\alpha_+ = 10°$, and R_i is set to be the minimum distance from among all the distances between x_i and intersection points of sampling rays shot from x_i. These two parameters R_+ and R_- control the maximum and minimum distances, respectively, between any pairs of gathering points, and are specified by the user according to accuracy requirements. Each gathering point is therefore valid only in a sphere with the radius $max(\sqrt{1-\cos(\alpha_+)}, max(min(R_i/2, R_+), R_-))$, which is called the effective radius of the gathering point.

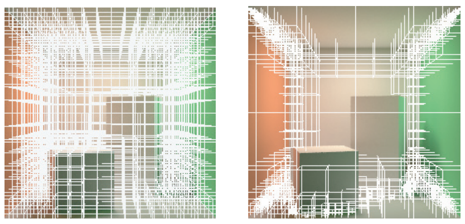


Figure 7. The finest level, subdivided uniformly (a) and adaptively (b).

We then shoot gathering rays from each gathering point. The intersection point of a gathering ray is stored as the index of the corresponding grid-cell in an appropriate level, and the index of the reverse direction of the gathering ray. The data size for each gathering ray is only 7 bytes (5 bytes for grid-cells and 2 bytes for direction).

5.3. Calculation of HVDS

The determination of the resolution of the grid-cells of the finest level is crucial to the final gathering. Finer resolution grid-cells are required in places near edges or corners while coarser resolution may be sufficient for other places. Our method determines the appropriate sizes of grid-cells efficiently as follows. Initially, the scene is subdivided uniformly. Since the projected area for each gathering ray is calculated during the pre-computation of gathering rays, the minimum projected area at each grid-cell can be obtained. We maintain an octree that covers the entire scene and adaptively subdivide the octree based on the projected areas of gathering rays. We show examples of grid-cells uniformly subdivided and adaptively subdivided in Figure 7. After we determine the resolution of the finest level, we construct the hierarchy by grouping the neighbor grid-cells.

6. Rendering

In this section, we describe the recalculation of the photon energies, firstly. Secondly, we describe how to store photon energies in the HVDS. Lastly, final gathering using the HVDS is described.

6.1. Recalculating photon energies

Energies of photons are recalculated when light sources are moved or the materials of the surfaces are changed. We first explain the calculation in the case where light sources move.

First, we calculate the energy transport between light sources and entry points. Let P_L be a sample point on a light source (Figure 8). n_L and $n_{i,0}$ are the normals at P_L

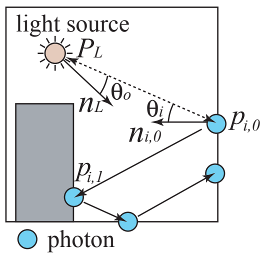


Figure 8. Calculating the energy of each photon.

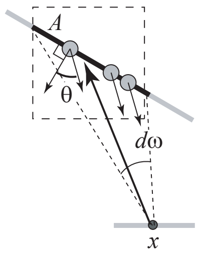


Figure 9. Energy of a grid-cell contributing to a calculation point.

and $p_{i,0}$, respectively. The energy E, assigned to the entry point $p_{i,0}$ from P_L is given by,

$$E = L_e(P_L \rightarrow p_{i,0})G(P_L \leftrightarrow p_{i,0})\pi f_s(P_L \rightarrow p_{i,0} \rightarrow p_{i,1}). \tag{8}$$

The factor π appears since we have selected the direction $(p_{i,0} \rightarrow p_{i,1})$ according to the cosine term. $G(P_L \leftrightarrow p_{i,0})$ can be computed as follows. Let r be the distance between P_L and $p_{i,0}$, and dA_i be the differential area related to $p_{i,0}$. Then the solid angle, $d\omega$, of $p_{i,0}$, viewed from P_L, is calculated from $d\omega = (\cos\theta_i/r^2)dA_i$. The geometry term is calculated by

$$G(P_L \leftrightarrow p_{i,0}) = V(P_L \leftrightarrow p_{i,0}) \cos\theta_o d\omega, \tag{9}$$

where $V(P_L \leftrightarrow p_{i,0})$ is the visibility function between P_L and $p_{i,0}$. For area light sources, points on the area sources are sampled and the calculations described above are performed per sample point. For point light sources, $\cos\theta_o$ is set to 1. Our method can deal with spot lights by modifying $L_e(P_L \rightarrow p_{i,0})$. For parallel lights, the direction from P_L to $p_{i,0}$ is set to the direction of the parallel light source, and both r and $\cos\theta_o$ are set to 1. For distant lighting, represented by an environment map, the incident radiances from the distant lighting are approximated by a set of parallel light sources.

Energies of photons $p_{i,j} (j \geq 1)$ are calculated by multiplying the pre-computed reflectances.

In the case where the material of a surface is changed, the reflectances of the photons on the surface are required to be recalculated. To recalculate the reflectance of $p_{i,j}$, we set the reflectance to $f_s(p_{i,j-1} \rightarrow p_{i,j} \rightarrow p_{i,j+1})/P_\sigma^\perp(p_{i,j-1} \rightarrow p_{i,j} \rightarrow p_{i,j+1})$, where $P_\sigma^\perp$ is the pdf used in pre-computation.

6.2. Storing photon energies

Firstly, we accumulate the photon energies into the finest level of the HVDS. To describe this process in more detail, let us assume that a set of photons within a single grid-cell have the same outgoing direction (Figure 9). Let the energy of each photon be E_i, and the area of the surface within the grid-cell be A. We simply store the accumulated energies (i.e., $\sum_i E_i$) in the grid-cell. The contribution of the photons seen from a calculation point x can then be calculated as follows. Let θ be the angle between the normal direction of the surface and the outgoing direction. The total energies of the photons per unit area within the grid-cells will be $\pi \sum_i E_i/(A\cos\theta)$, if the outgoing direction of photons was selected according to the cosine term (in a general case, $\sum_i E_i/(AP_\sigma^\perp(\theta)\cos\theta)$). Suppose that we have discretized the direction uniformly into N_{dir} direction clusters, then each direction cluster will have the solid angle of $4\pi/N_{dir}$. Therefore the total energy of the photons per unit area per solid angle is $N_{dir}\sum_i E_i/(4A\cos\theta)$. Let r be the distance between x and the center of the grid-cell, the solid angle $d\omega$ of the grid-cell viewed from x is given by $A\cos\theta/r^2$. Therefore, the energies leaving the grid-cell toward x is equal to $N_{dir}\sum_i E_i/4r^2$. Hence, the recorded energy in the grid-cell can be efficiently used during final gathering.

Next, we explain how to calculate the energies stored in coarser levels of the HVDS. It can easily be shown that by simply summing up the stored energies in the corresponding grid-cells in the finer level, we can obtain the energies that are to be stored in the coarser level, and the energies can be used in exactly the same way as described above.

6.3. Rendering using final gathering

In the rendering of the scene, our method first calculates the positions, normals and colors of the visible surfaces based on the concept of the deferred shading [18]. Then the direct illumination of the visible surfaces is calculated by using previous methods, and shadows are rendered by using a shadow mapping method. The indirect illumination of the visible surfaces is interpolated by the radiances due to indirect illumination at gathering points. Although radiance caching [14] can interpolate the indirect illumination of glossy surfaces more accurately than irradiance caching, we used irradiance caching since irradiance caching is more efficient and is sufficient for low frequency glossy surfaces. The outgoing radiance at each gathering point is calculated by summing the estimated incident radiance estimated using the hierarchical histogram estimation, multiplied by the BRDF of the gathering point.

7. Implementation on the GPU

In this section, we describe the overview of our GPU implementation, while details of the GPU implementation are described in Appendix. The rendering process consists of two stages. The first stage is the process in the object space as described below, and the second one in the screen space using the irradiance cache splatting (similar to photon splatting [15]). The first stage consists of the following five steps.

1. calculation of the energies of entry points $p_{i,0}$
2. calculation of the energies at the reflected locations $p_{i,j} (j \geq 1)$
3. storing the photon energies into the finest level of the HDVS
4. construction of coarser levels of the HDVS
5. calculation of outgoing radiances at gathering points

To calculate the energies at entry points, the calculation of the visibility function V is required. We use shadow mapping for efficient calculation.

The ideas to perform steps 3, 4, and 5 on the GPU are as follows. Since we consider only static scenes, the hierarchy of the volumetric data structure and the correspondences between each grid-cells, between photons and grid-cells, and between gathering rays and grid-cells do not change. Therefore, by carefully reordering the process, we can rearrange the rendering process to a fashion that is suited for sequential processing using GPUs.

8. Results

We show some results rendered by using our method and compare them with images rendered using a pure Monte Carlo solution. We used an NVIDIA GeForce 7800 GTX and 3.0 GHz PentiumD CPU with 2GB RAM for all the results shown in this section.

We show the rendered results of four scenes: Box, Sponza, Tree, and Room. The numbers of triangles in these scenes and the rendered image sizes are shown in Table 1. The rendered images, using the CPU implementation of our method, are shown in Figure 10.

In the Box scene, a light source is placed in the upper backward side of the scene. The front of the glossy teapot is illuminated due to the indirect illumination. The Sponza scene is lit by a point light source located in the back corridor, and most parts of the scene, such as the pillars and the upper part of the arch seen through the pillars, are illuminated due to indirect illumination. The Tree scene is complex in visibility, and contains a glossy bunny. A light source is placed above the tree so that the leaves, the floor and the bunny are illuminated mostly by indirect illumination. The rendered results of the Room scene show an example of lighting design.

The RMS errors measured with respect to the pure Monte Carlo solution are between 1.3% to 5.7%. As shown in these figures, the images rendered by using our method are indistinguishable from the images rendered by using pure Monte Carlo solution.

Table 1. Results.

Scene	Box	Sponza	Tree	Room
#Tri.	2,914	76,154	141,287	121,696
Image Size	512 × 512	640 × 480	640 × 480	640 × 480
T_{di}	6 ms	15 ms	30 ms	28 ms
T_{pe}	11 ms	15 ms	23 ms	22 ms
T_{hvds}	11 ms	12 ms	112 ms	107 ms
T_{fg}	5 ms	27 ms	142 ms	162 ms
T_{splat}	76 ms	122 ms	281 ms	302 ms
T_{tot}	109 ms	191 ms	588 ms	621 ms
T_{light}	109 ms	191 ms	588 ms	621 ms
T_{view}	87 ms	164 ms	453 ms	492 ms
T_{mat}	1271 ms	1310 ms	1507 ms	1911 ms
$T_{precomp}$	1h30m	2h	2h40m	2h20m
Memory	7.5MB	25.1MB	75.0MB	84.7MB
1 light	9.1 fps	5.2 fps	1.7 fps	1.6 fps
2 lights	8.6 fps	4.8 fps	1.5 fps	1.3 fps
4 lights	8.2 fps	4.3 fps	1.2 fps	1.1 fps
8 lights	7.2 fps	3.6 fps	1.0 fps	0.9 fps
Gather pts.	2k	17k	36k	39k
#Gthr rays	291k	2M	8M	8M
#Photons	426k	638k	730k	1.4M

The parameters for rendering these scenes are shown in Table 1. The resolutions of the discretized directions are all set to be 128 for these scenes.

Next, let us show the results rendered by using the GPU implementation and discuss the details of the computational time. Results are summarized in Table 1. In the case where the materials are fixed, the rendering process consists of the calculation of direct illumination (shown as T_{di} in Table 1), recalculation of photon energies on the GPU (T_{pe}), storing photon energies into the HVDS (T_{hvds}), the final gathering (T_{fg}), and irradiance splatting (T_{splat}). The total time to render a single frame is shown as T_{tot}. Note that most time during rendering is spent on irradiance splatting. *Memory* shows the required memory to store the pre-computed photon paths and gathering rays for each scene. T_{light}, T_{view}, and T_{mat} show the times to render a single frame when we move a light source, the viewpoint, and modify a material, respectively.

In frames where the light sources remain fixed, we do not have to recalculate the photon energies or re-store the photon energies into the HVDS. In case where materials are changed, the reflectance of each photon and the material information of each gathering point have to be updated, resulting in the increase of the computational time. We also show the rendering performance as we vary the number of light sources in Table 1. For each light source, we use the same shadow map to calculate both direct illumination and the energies at the entry points. Therefore, the computational time does not increase drastically with the increase in the number of light sources. The performance gained by running the rendering process on the GPU instead of CPUs is about an order of acceleration. We show some captured images from our GPU implementation in Figure 11.

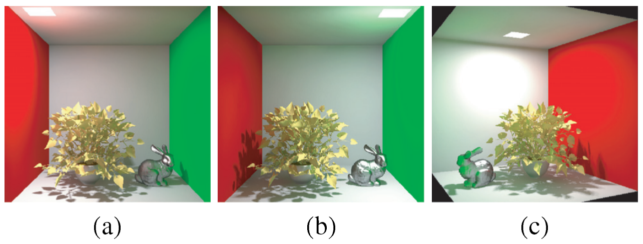


Figure 11. Rendered results of the Tree scene using our method. (a) and (b): moving the light source, (c): moving the viewpoint.

9. Limitations and Discussion

Our method makes several approximations and limitations to achieve interactive rendering. Firstly, since we discretize the direction of photons and gathering rays, we can deal only with low-frequency BRDFs. Though it is possible to deal with materials with sharp gloss by increasing the number of discretized directions, the required memory will increase. A GPU-based implementation is not possible in these cases. The minimum number of discretized directions N_{dir} needed to deal with a glossy surface with certain cosine exponent n can be decided by considering that a single discretized direction should at least be smaller than the range of the glossy lobe. The relationship between N_{dir} and n is given by $n \leq 1/\tan^2(\pi/N_{dir})$. Practically, N_{dir} should be larger, and in our experiments, we found that we can deal with Phong BRDFs with cosine exponent up to 15 with 128 directions. We believe the discretization in the Cartesian coordinates will not be a critical problem since the resolution of the finest level of the HVDS is adaptively determined according to the scene complexity.

Secondly, when using a grid-cell in a coarser level, since the energies in the grid-cell are simply the sum of the energies in corresponding grid-cells in the finer level, occulusion is ignored which may result in errors (Figure 12(a)). We believe we can solve this problem by replacing the gathering ray collecting the energy from the grid-cell in a coarser level with several gathering rays collecting the energies from the grid-cells in the finer level (Figure 12(b)). We repeat this process until each ray collects energy from a grid-cell where the occlusion problem can be ignored.

Thirdly, we ignore shadows due to objects representing light sources.

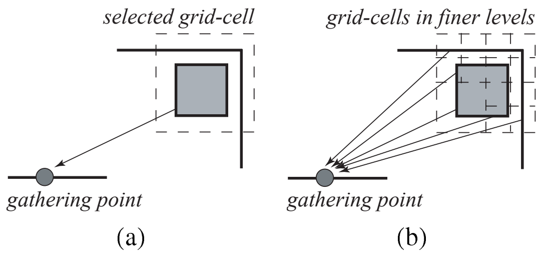


Figure 12. Using the coarser level will cause problems (a). Replace gathering rays to gather from finer levels (b).

10. Conclusion and Future Work

We have proposed two techniques aimed for the fast computation of global illumination: a path pre-computation algorithm and the hierarchical histogram estimation. By pre-computing the paths, energies of photons can be recalculated very quickly, without involving ray tracing calculations. By directly using the radiance recorded in the HDVS, we can perform final gathering without any nearest neighbor searching, but only a few array accesses.

By using the proposed techniques, we achieved interactive rendering of complex scenes with global illumination. Our method can deal with any type of local light sources and can change the materials of the surfaces interactively, therefore can be applied to lighting design.

In future work, we aim to achieve the interactive rendering of dynamic scenes, where objects in the scene move. In such scenes, a part of the light paths changes and needs to be recalculated. If the object moves rigidly, the density of the entry points on the surface of the object does not change. Therefore, photon paths, which are changed due to the movement of the object, are re-sampled. The gathering points and the gathering rays can be re-sampled according to [25], which uses the temporal coherence and changes a portion of gathering rays one by one.

References

[1] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Transactions on Graphics*, 24(3):1108–1114, 2005.

[2] P. Bekaert, M. Sbert, and J. Halton. Accelerating path tracing by re-using paths. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 125–134. Eurographics Association, 2002.

[3] P. H. Christensen and D. Batali. An irradiance atlas for global illumination in complex production scenes. In *Rendering Techniques*, pages 133–142, 2004.

[4] M. F. Cohen and D. P. Greenberg. The hemi-cube: a radiosity solution for complex environments. In *SIGGRAPH '85*, pages 31–40, 1985.

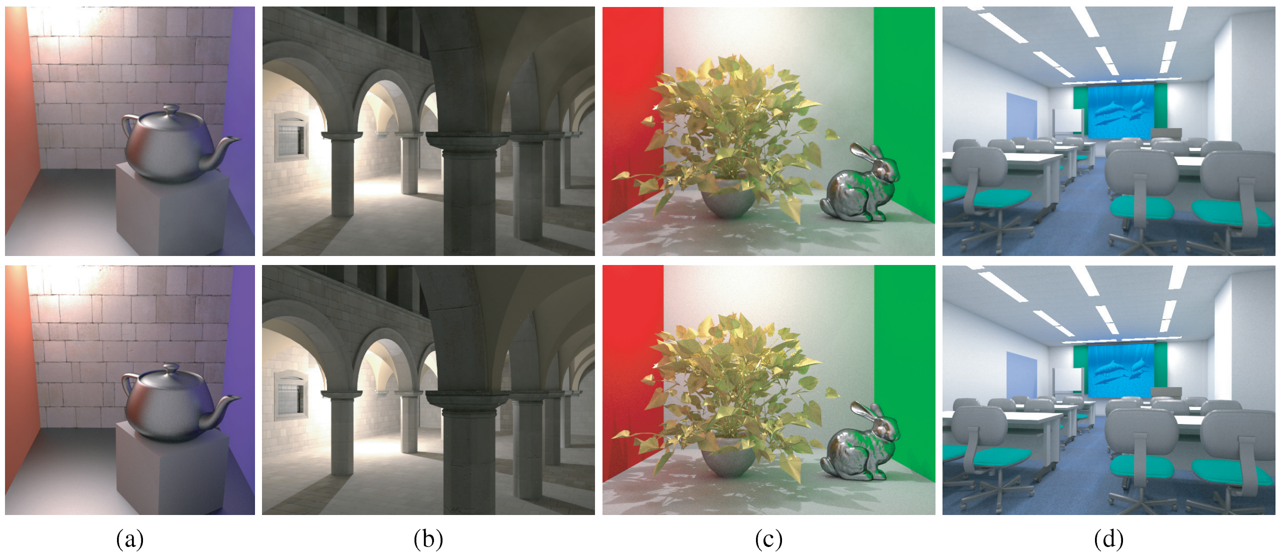


Figure 10. The top row shows the images rendered with our method, and the bottom row shows the images rendered with pure Monte Carlo. (a) Box, (b) Sponza, (c) Tree, and (d) Room.

[5] C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 2007, to appear.

[6] K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel. Interactive global illumination using selective photon tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 25–36, 2002.

[7] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 197–206, New York, NY, USA, 1991. ACM Press.

[8] M. Hašan, F. Pellacini, and K. Bala. Direct-to-indirect transfer for cinematic relighting. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1089–1097, New York, NY, USA, 2006. ACM Press.

[9] H. Igehy. Tracing ray differentials. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 179–186, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[10] K. Iwasaki, Y. Dobashi, F. Yoshimoto, and T. Nishita. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proc. Eurographics Symposium on Rendering 2007*, pages 35–44, 2007.

[11] H. W. Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, 1996.

[12] J. T. Kajiya. The rendering equation. In *SIGGRAPH '86*, pages 143–150, 1986.

[13] A. W. Kristensen, T. Akenine-Moller, and H. W. Jensen. Precomputed local radiance transfer for real-time lighting design. In *SIGGRAPH '05*, pages 1208–1215, 2005.

[14] 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.

[15] F. Lavignotte and M. Paulin. Scalable photon splatting for global illumination. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 203–210, New York, NY, USA, 2003. ACM Press.

[16] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 376–381, New York, NY, USA, 2003. ACM Press.

[17] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *SIGGRAPH '85*, pages 23–30, 1985.

[18] M. Pharr. *GPU Gems 2*. Addison-Wesley, Upper Saddle River, NJ, 2005.

[19] M. Sbert, L. Szecsi, and L. Szirmay-Kalos. Real-time Light Animation. *Computer Graphics Forum*, 23(3):291–299, 2004.

[20] A. Scheel, M. Stamminger, and H.-P. Seidel. Grid based final gather for radiosity on complex clustered scenes. *Computer Graphics Forum*, 21(3):547–556, 2002.

[21] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA, 2002. ACM Press.

[22] L. Szécsi, L. Szirmay-Kalos, and M. Sbert. Light animation with precomputed light paths on the gpu. In *Graphics Interface 2006*, pages 187–194, 2006.

[23] E. Tabellion and A. Lamorlette. An approximate global illumination system for computer generated films. *ACM Transactions on Graphics*, 23(3):469–476, 2004.

[24] N. Tamura, H. Johan, B.-Y. Chen, and T. Nishita. A practical and fast rendering algorithm for dynamic scenes using

adaptive shadow fields. *The Visual Computer*, 22(9):702–712, 2006.

[25] T. Tawara, K. Myszkowski, and H.-P. Seidel. Exploiting temporal coherence in final gathering for dynamic scenes. In *Computer Graphics International (CGI '04)*, pages 110–119, 2004.

[26] E. Veach. *Robust monte carlo methods for light transport simulation.* PhD thesis, Stanford University, 1998. Adviser-Leonidas J. Guibas.

[27] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88*, pages 85–92, 1988.

[28] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum. Precomputed shadow fields for dynamic scenes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1196–1201, New York, NY, USA, 2005. ACM Press.

Appendix Details of GPU Implementation

For the sake of the simplicity, the resolutions of the grid-cells in the base level are assumed to be the same and the details of step 4 are explained before those of step 3.

Calculation of energies at entry points: Initially, a texture storing the positions of the entry points $p_{i,0}$ is prepared. Then sample points for the calculation of the visibility function V are allocated on light sources. The depth values seen from each sample point on a light source are stored in each texture. The visibility function V is calculated by comparing the depth stored in the texture and the depth of entry points on a fragment program. At the same time, the energies of the entry points are calculated from Eq.(8) and are stored as a texture.

Calculation of photon energies: To calculate the energies of photons $p_{i,j}$ on the GPU, $N_{max}+1$ textures $T_0^p, T_1^p, ..., T_{N_{max}}^p$ are prepared to store energies of photons, where N_{max} is the maximum number of bounces (in our case, $N_{max} = 3$). The i-th pixel of the j-th texture T_j^p stores the energy of the photon $p_{i,j}$. We call these textures *photon energy maps*. The j-th texture T_j^p is calculated by multiplicative blending of the $(j-1)$-th texture T_{j-1}^p and the texture that stores the reflectances at $p_{i,j}$ in the i-th pixel.

Construction of the HVDS: The HVDS at level l (coarser resolution) is constructed by using the HVDS at level $l-1$ (finer resolution). To represent the HVDS, which is a tree structure, on the GPU, we use textures, each corresponding to each level of the HVDS. To construct the HVDS efficiently, we re-order the texture data. Let T_l^h be the texture corresponding to the HVDS at level l. To construct the texture T_l^h, we first reorder the texture T_{l-1}^h so that the pixels of T_{l-1}^h, corresponding to the pixel of T_l^h, are packed as shown in Figure 13. And for each pixel of T_l^h, the addresses of the corresponding pixels of T_{l-1}^h are stored as textures.

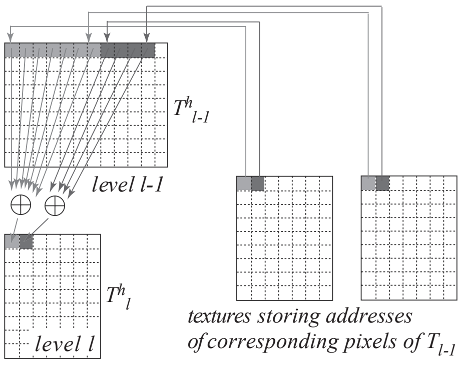

T^h_{l-1}
level l-1
T^h_l
level l
textures storing addresses of corresponding pixels of T_{l-1}

Figure 13. Construction of the hierarchical histogram on the GPU.

Storing energies of photons into the base level: The operations for this step are similar to those for the construction of the hierarchical histograms. We prepare a texture T_0^h, corresponding to the base level, where each pixel of the texture stores the accumulated energies of the corresponding component of the base level. For each component of the base level, the indices of the photons that are stored in the component are stored as a texture, called the *photon index map* in Figure 14. For each component of the base level, the addresses of the photon index map are stored as textures.

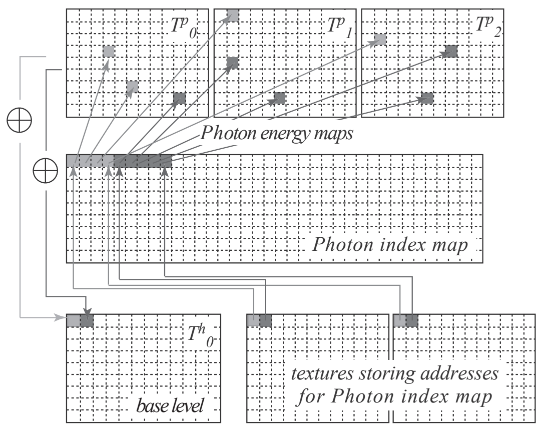


Figure 14. Storing photon energies using GPUs.

Calculation of indirect illumination on gathering points: The energies of the gathering points are calculated by referring to the data of the HVDS. This calculation is performed in a similar way to storing the energies of photons in the base level. We create a texture that stores the energies of the gathering points and each pixel of the texture corresponding to each gathering point. Then we perform splatting of gathering points on a vertex program by using the texture storing the energies of gathering points. In this step, the texture that stores the material color of the object surface is mapped onto the splats with multiplicative blending.