Acceleration of Photon Mapping: Adaptive Sampling of Irradiance Cache by Components

Ko-Fan Lu, National Tsing Hua University
Chun-Fa Chang, National Taiwan Normal University

Abstract— **Photon mapping is a two-pass method of ray tracing that makes the computation of caustics and soft indirect illumination relatively fast. Besides, the irradiance cache is a method to further speed up the computation of indirect illumination in a Monte Carlo ray tracer. This paper presents a method that uses the concept of irradiance cache in multi-resolution to cache the irradiance of each light component in photon maps. Our method can not only make the computation more efficient but also preserve the details of scene.**

I. INTRODUCTION

Ray tracing is a method which is often used for rendering the global illumination, but it demands heavy computation on the rendering of high resolution images and complex lighting effects.

In photon maps [10], Jensen proposed a two-pass method to increase the efficiency of ray tracing. In the first pass, the method stores the irradiance from light source to objects into a photon map, and then in the second pass queries the photons for gathering the final intensity. By this two-pass ray tracing, the computation of indirect illumination and caustics effect become more efficient.

Irradiance Cache is also a method for speeding up the computation of indirect illumination in a Monte Carlo ray tracer introduced by Ward et al.[17][18][19]. Because it is a method of caching and reusing irradiance values on Lambertian surface, it can work well in the rendering step of the two-pass photon-mapping algorithm. Most of previous works combine the above two methods together, which include caching and interpolating the irradiance on the surfaces of objects in 3D space, only for the indirect illumination. It might cause extra computation due to the occlusion of object, and cannot increase the efficiency of computing other light components.

We proposed a method which applies the concept of irradiance cache but we interpolate the contribution from rays that initiate from the eyes. Because two pixels on the rendered image might have the same reflection, we cache the irradiance from eye rays in low resolution, and interpolate the irradiance in high resolution. This mechanism is used not only for computing indirect illumination, but also used for any other light components. Therefore we can speed up the efficiency furthermore, and still preserve the details of the scene.

In the following sections, we first introduce the concept of photon mapping and irradiance cache and discuss related works in Section 2. Section 3 describes our method in detail, and the results are presented in Section 4. Discussion, conclusion and future works are then presented in Sections 5 and 6.

II. BACKGROUND

The method we proposed is primarily based on photon mapping [9][10][11] and irradiance cache [17][18][19][20]. Besides, the concept of interpolation in our method, which querying the cached irradiance of the nearby pixels, is inspired by adaptive ray tracing [14][5]. In this section, we first introduce photon mapping, irradiance cache and adaptive ray tracing briefly, and then discuss the previous works which also accelerate the computation of photon mapping and ray tracing.

A. Photon Mapping

There are two passes when tracing the irradiance of rays with photon maps.

In the first pass, the photon is emitted from light source in the scene. Each photon is traced through the scene using a method similar to path tracing. Every time a photon hits a surface it is stored within the photon map and a Monte Carlo method called Russian roulette [7] is used to determine the photon is absorbed or reflected. Typically, there are two photon maps for a scene, one is global photon map, and the other is caustics map. The photons are stored in a balanced kd-tree which is a compact and efficient data structure.

After constructing the photon maps, the final image is rendered using Monte Carlo ray tracing in the second pass. Each sample consists of tracing a ray from the eye through the pixel into the scene. For each pixel, the scene is ray traced until the closest surface of intersection is found.

B. Irradiance Caches

The irradiance caching method was introduced by Ward et al. [17] in 1994 as a method for speeding up the computation of indirect illumination. It is a method for caching and re-using with interpolation irradiance values on Lambertian surface. The irradiance at a location x, on a diffuse surface is computed by sampling the incident radiance above x. When computing a new irradiance value at x_i, we first look at the previously computed irradiance values. For each of those values we compute a weight w_i that tells us if we can use the value for interpolation, the weight w_i is computed as

$$w_i(x,\vec{n}) = \frac{1}{\varepsilon_i(x,\vec{n})}, \quad (1)$$

where $\varepsilon_i(x)$ is an estimate of the amount of change in the irradiance from x_i to x. One of the two possible changes is a change to the surface location and another is a change in the surface orientation, so the weight w_i can be modified as

$$w_i = \frac{1}{\frac{\|x_i - x\|}{R_0} + \sqrt{1 - \vec{n}\cdot\vec{n}_i}}. \quad (2)$$

To compute an estimate of the irradiance at x, we compute a weight for all the previously computed irradiance values.

Ward [18] suggested a user-controlled parameter that is proportional to the maximum allowed error on the estimate τ, and for the irradiance values where $w_i < \tau$ we get

$$E(x,\vec{n}) \approx \frac{\sum_{i,wi<\tau} w_i(x,\vec{n})E_i(x_i)}{\sum_{i,wi<\tau} w_i(x,\vec{n})}. \quad (3)$$

If there is no previously computed irradiance value with a sufficiently high weight, we have to compute a new one.

C. Adaptive Ray Tracing

A ray tracing method over a regular pixel grid leads to redundant computations on the one hand, and is prone to aliasing artifacts on the other. For anti-aliasing, Cook [5] used two levels of sampling density, to concentrate samples where they are needed most. This is also a ray tracing method which can be used in parallel easily, the idea is proposed by Notkin et al. [14]

There are several previous works [2][3][4] using the irradiance cache method to accelerate the computation of ray tracing. They follow the idea introduced by Ward, caching and interpolating on the surface in 3D space. Christensen [3] even applied the idea on photon mapping method. Different from the previous work, our method uses the concept of adaptive ray tracing to re-use the cached irradiance from the view point of the pixels on the scene.

III. ALGORITHM

A. Light Components

According to the rendering equation, the incoming radiance can be split into a sum of several components from the rendering equation. We used the four components the same as Henrik proposed in photon mapping which are shown in the figure 1.

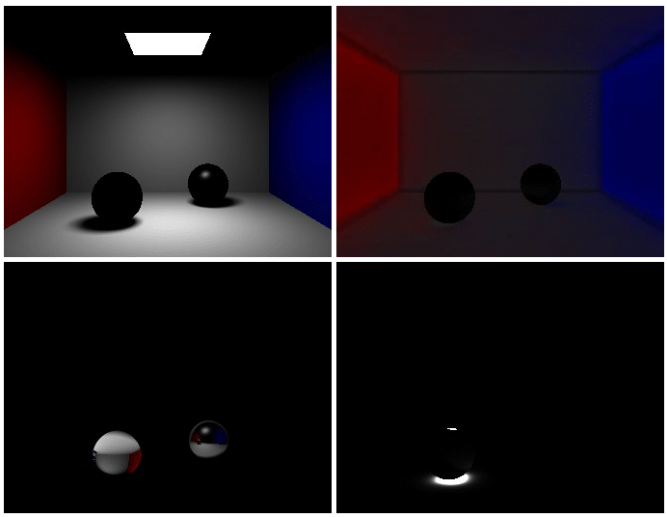

Fig. 1 The four light components: direct illumination (left up), indirect illumination (right up), specular reflection (left down), and caustics.

The rendering equation is described as

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + L_r(x,\vec{\omega}), \quad (4)$$

which computes the outgoing radiance L_o at a given surface location x with a given direction $\vec{\omega}$. The reflected radiance is computed by the integral

$$L_r(x,\vec{\omega}) = \int_\Omega f_r(x,\vec{\omega}',\vec{\omega})L_i(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n})d\vec{\omega}', \quad (5)$$

which are combined with the BRDF f_r and the incoming radiance L_i.

The BRDF often combined with two components, one is diffuse component which describes the smooth illumination, and the other is specular component which describes the sharp illumination. Therefore, the BRDF is split into the sum of two terms,

$$f_r(x,\vec{\omega}',\vec{\omega}) = f_{r,S}(x,\vec{\omega}',\vec{\omega}) + f_{r,D}(x,\vec{\omega}',\vec{\omega}). \quad (6)$$

The same as the incoming radiance L_i can be split into the sum of three components

$$L_i(x,\vec{\omega}') = L_{i,l}(x,\vec{\omega}') + L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}'), \quad (7)$$

where $L_{i,l}(x,\vec{\omega}')$ is direct illumination from the light sources, $L_{i,c}(x,\vec{\omega}')$ is caustics, the indirect illumination from the light sources via specular reflection or transmission, and $L_{i,d}(x,\vec{\omega}')$ is indirect illumination from the light sources that has been reflected diffusely at least once.

Finally, we combine the classified BRDF and the incoming radiance and then split the reflected radiance into a sum of four integrals which are presented as four light components; the four integrals are described as follows:

$$
\begin{aligned}
&L_r(x,\vec{\omega}) \\
&= \int_\Omega f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n})d\vec{\omega}' \\
&= \int_\Omega f_r(x,\vec{\omega}',\vec{\omega}) L_{i,l}(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n})d\vec{\omega}' \\
&+ \int_\Omega f_{r,S}(x,\vec{\omega}',\vec{\omega})(L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n}))d\vec{\omega}' \\
&+ \int_\Omega f_{r,D}(x,\vec{\omega}',\vec{\omega}) L_{i,c}(x,\vec{\omega}')d\vec{\omega}' + \\
&+ \int_\Omega f_{r,D}(x,\vec{\omega}',\vec{\omega}) L_{i,d}(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n}))d\vec{\omega}'
\end{aligned} \tag{8}
$$

B. *Multi-resolution Sampling*

Different with caching and interpolating irradiance on the object surface in 3D space, our method caches the irradiance of the low-resolution samples and then interpolates for rendering high-resolution scene in 2D space. Figure 2 shows the sampling step in the work flow of our method.

The sampling in low-resolution is simply with random. But how sparse of the samples is a considerable problem. Because too many samples will cause large computation time in low-resolution. Nevertheless, if there are not enough samples to be the candidates of interpolation, not only the rendering quality decreases, the pixel that does not have suitable candidate for interpolating will compute the irradiance by tracing the whole eye ray iteratively. As computing each pixel in low-resolution, it will not accelerate apparently in high-resolution caused by too few samples to be the candidates of interpolation.

We sample in one-fourth of the region for searching the candidates of interpolation randomly to assure there are enough samples for candidates also prevent huge computation time for low-resolution.

C. *Cache and Interpolation*

After sampling in the low-resolution, we construct a cache-map to store the irradiance of four components and surface information, wherein the surface information is first hit by the eye rays includes location and direction. The structure of cache map is as follow:

```
Cache map[Image Width][Image Height]
{
        // The surface information of first intersection.
        Vector Position;   // Location
        Vector Normal;     // Orientation

        // The irradiance of light components
        RGB Ld;            // Direct illumination
        RGB Li;            // Indirect illumination
        RGB Ls;            // Specular reflection
        RGB Lc;            // Caustics

        // Check the pixel is computed or not
        Boolean Flag;
}
```

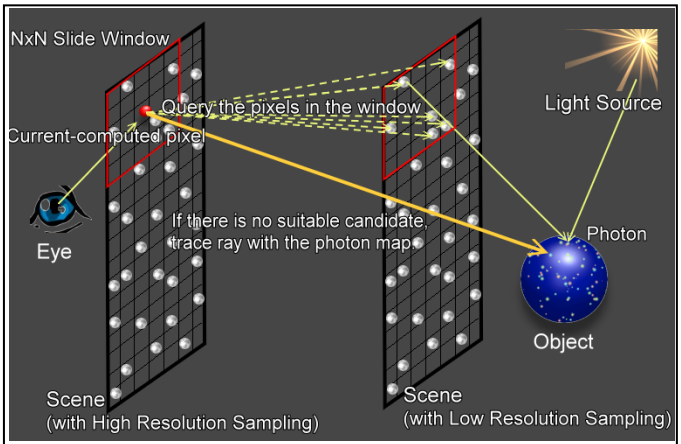

Fig. 2 The work flow of our method, where the white spot on the scene is the samples in low-resolution and cache the irradiance computed from eye ray to the photon stored on the surface. The red spot is the pixel we computed currently, and the red line marks a region where the computed pixels in this region will be the candidates for interpolation. If there is no suitable candidate, the pixel must be computed by ray tracing with photon map, which was show as the thicker yellow path.

Then we compute the rest pixels in high-resolution with interpolation. We check every computed pixel in a region centered with the pixel we compute currently. Different light components should have different condition to check if they should be the candidate or not. The direct, indirect (diffuse), and specular component all are checked by the location relationship
of the candidates and the pixel computed currently. Although the choose mechanism is the same, they will have different thresholds. Because the region of caustics is often small, we interpolate the pixel that all the irradiances of candidates in the region centered with it are zero. Where the conditions of the four light components are described as follows:

$$w_{i,d} < \tau_d \quad \text{for direct illumination,} \tag{9}$$

$$w_{i,i} < \tau_i \quad \text{for indirect illumination,} \tag{10}$$

$$w_{i,s} < \tau_s \quad \text{for specular reflection,} \tag{11}$$

$$\text{and } \sum L_{i,c}(x_i) = 0 \quad \text{for caustics,} \tag{12}$$

where the thresholds τ_d , τ_i and τ_s are all user-controlled, and $L_{i,c}(x_i)$ is the irradiance of candidate in the region centered with the current-computed pixel.

If the pixel satisfies the condition separately with four components, the irradiance of each component stored in the cache-map is multiplied by the weight calculated by location and position, and the weight equation is described in equation (2).

The sum of irradiance of all candidates is divided with the sum of weight as the equation (3). The illumination of each light component was calculated separately by the above equations. Then we sum up the four irradiances of each light component to compute the final illumination.

IV. RESULTS

This method has been implemented on a 2.66GHz Pentium PC running Windows XP with 3.48GM RAM.

The first scene we tested is "Cornell box" with a metal and a glass sphere, and there is only one light source. Emitting 1,000,000 photons and resulting in 2,500,000 photons stored in the global photon map and 300,000 photons in the caustic photon map. The first pass of construction of photon maps is around 17 seconds. The entire statistics of rendering time are visualized in Figure 3, and the rendering time in detail is arranged in table 1.

In Figure 3, we aimed to save the computation time of direct and indirect illumination, which are the two primary part of rendering time as we can see in Figure 3. The right bars of every image size are the results after using cache mechanism. According to the two trend lines, our method makes the rendering time increase mildly with the image size. Figure 4 are the two results where the left one is rendering without cache, and the right one is with cache. Table 1 and Figure 4 show that our method can save 15% to 30% of the rendering time and preserve the detail of the image.

We add another light source from different direction and a more complicated model, a glass bunny, in the "Cornell box" to build the second tested scene "Rabbit". Emitting 1,000,000 photons each light, and resulting in 4,000,000 photons in the global photon map and 550,000 photons in the caustic photon map. This scene takes about 49 seconds to construct the photon map. We also save around 30% to 70% of the rendering time of the scene increasing with the image size which can be shown in table 2. The Figure 5 shows the statistics of rendering time, and the rendering images are shown in Figure 6.

According to the two tested scenes, we calculate the average time per pixel of rendering each component: direct illumination takes 2.37 ms/pixel, 88.92% of rendering time; indirect illumination takes 0.058 ms/pixel which is 2.16% of rendering time; specular reflection takes 0.225 ms/pixel, 8.44% of rendering time, and the caustics spend 0.013 ms/pixel, which is 0.48% of rendering time.

V. DISCUSSION AND CONCLUSION

Our method can make the two-pass ray tracing with photon maps more efficient and still preserve the important information in the scene. The key point is that we cache the illumination in low-resolution and then reuse them for interpolation in high-resolution separately with light components. But there is still trade-off between the quality and performance according to the threshold of each light component. Therefore, how to find an appropriate threshold quickly is difficult, which will depend on the complexity of the scene and the object.

The method we proposed will be well-performed for super-sampling because of the efficient and automatic interpolation saving most of time when rendering high-resolution image. This method can also be combined with other acceleration mechanisms which cache the irradiance on the surface of object, and further speed up with multiple accelerated steps.

VI. FUTURE WORK

We will improve the sampling method separately with light components for accelerating the computation time in low-resolution, because of taking caustics component as example, there will be only a small region for caustics illumination in general, so does the specular component. Therefore, we should sample densely in those regions and relatively more sparsely in other parts of the scene.

We will also add other kind of light component to reduce the problems such as the shadow caused by direct illumination has a noisy curve after interpolating. If there are other considerable reflection properties, they should be separated into different component, too.

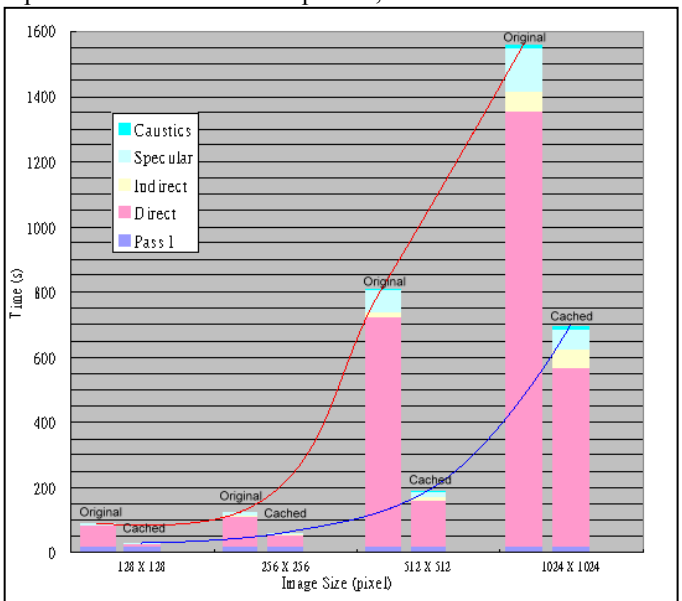

Fig. 3 The statistics of rendering time of Cornell Box. The four colors in order of top to bottom represent the four light components: caustics, specular reflection, indirect illumination, and direction. The last violet part is the time of constructing photon maps. In each section of image size, the left bar linked with red line is the rendering time without cache and interpolation, and the right bar linked with blue line is with cache and interpolation.

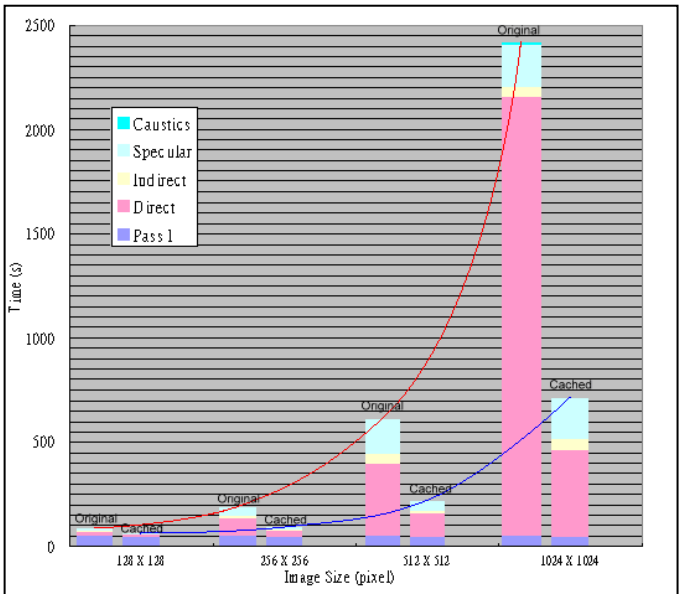

Fig. 4 The statistics of rendering time of Rabbit Scene. The four colors in order of top to bottom represent the four light components: caustics, specular reflection, indirect illumination, and direction. The last violet part is the time of constructing photon maps. In each section of image size, the left bar linked with red line is the rendering time without cache and interpolation, and the right bar linked with blue line is with cache and interpolation.

TABLE I
STATISTICS OF REDERING THE SCENE "CORNELL BOX"

	128 X 128		256 X 256		512 X 512		1024 X 1024	
	original	cached	original	cached	original	cached	original	cached
Caustics	0.29	0.25	0.77	0.65	3.22	2.71	12.47	11.20
Specular	5.97	0.65	9.67	3.92	67.73	15.01	136.24	61.43
Indirect	2.71	1.07	3.94	3.65	14.34	14.29	58.24	58.01
Direct	65.35	11.20	95.46	33.08	719.26	138.67	1364.48	545.32
Pass 1	17.36	17.21	17.36	16.82	17.06	17.36	17.12	17.34
Total time	89.84	29.60	126.36	58.96	821.85	187.81	1588.54	693.33
Saved time (%)	27.26%		24.25%		12.28%		14.10%	

TABLE II
STATISTICS OF REDERING THE SCENE "RABBIT"

	128 X 128		256 X 256		512 X 512		1024 X 1024	
	original	cached	original	cached	original	cached	original	cached
Caustics	0.11	0.11	0.24	0.15	1.00	0.19	1.57	0.20
Specular	10.68	2.88	43.00	12.69	168.13	46.23	203.56	194.92
Indirect	2.89	1.27	12.63	3.69	49.37	14.27	52.10	57.24
Direct	23.84	8.13	87.34	27.60	352.92	108.28	2144.61	409.20
Pass 1	49.35	49.85	49.42	49.59	49.42	49.60	49.40	49.23
Total time	86.87	62.24	192.84	93.72	620.84	218.56	2461.243	710.79
Saved time (%)	28.35%		51.40%		64.80%		71.12%	

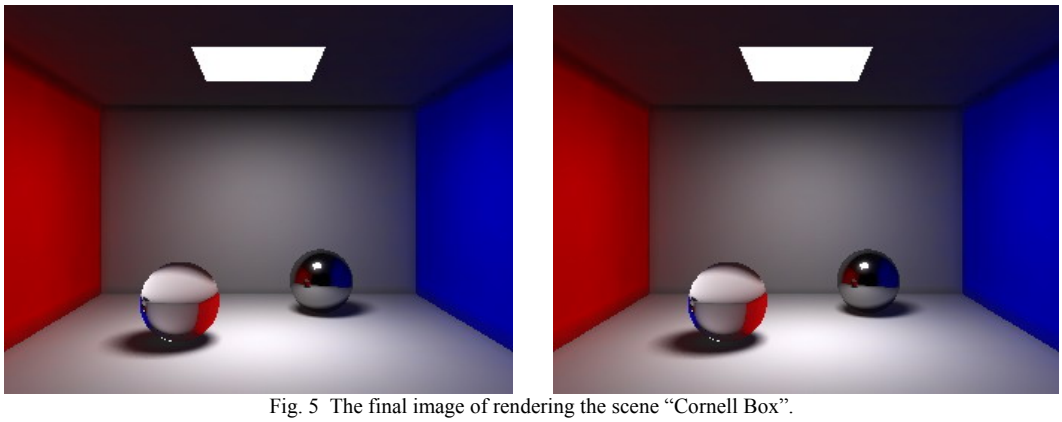

Fig. 5 The final image of rendering the scene "Cornell Box".
The left image in rendered by the original photon mapping, and the right image is rendered by our method.

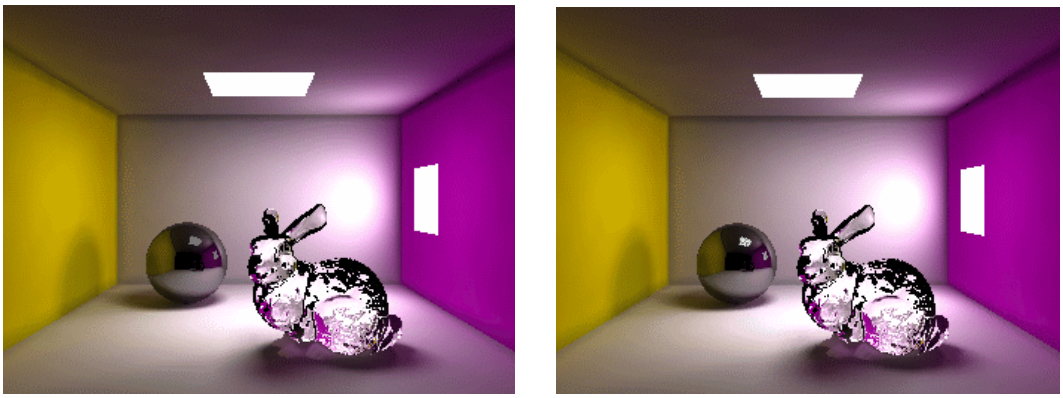

Fig. 6 The final image of rendering the scene "Rabbit".
The left image in rendered by the original photon mapping, and the right image is rendered by our method.

ACKNOWLEDGMENT

We thank the member of IBR Lab at National Tsing Hua University and Chuan-Chang Wang for giving support to our experiments. This work is supported in part by National Science Concil, Taiwan, under the grant NSC 97-2220-E-003-001.

REFERENCES

[1] Kavita Bala, Julie Dorsey, and Seth Teller, "Radiance interpolants for accelerated bounded-error ray tracing," *ACM Transactions on Graphics (TOG),* v.18 n.3, p.213-256, July 1999

[2] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner, "A progressive multi-pass method for global illumination," *ACM SIGGRAPH Computer Graphics,* v.25 n.4, p.165-174, July 1991

[3] Per H. Christensen, "Faster photon map global illumination," *Journal of Graphics Tools,* v.4 n.3, p.1-10, 1999

[4] Steven Collins, "Adaptive splatting for specular to diffuse light transport," *In Fifth Eurographics Workshop on Rendering,* pp.119-135, June 1994.

[5] Robert L. Cook, "Stochastic sampling in computer graphics," *ACM Transactions on Graphics (TOG),* v.5 n.1, p.51-72, January 1986

[6] Robert L. Cook, Thomas Porter, and Loren Carpenter, "Distributed ray tracing," *ACM SIGGRAPH Computer Graphics,* v.18 n.3, p.137-145, July 1984

[7] Arvo James and David Kirk, "Particle Transport and Image Synthesis," *Computer Graphics,* v.24 n.4, pp. 53-66, 1990

[8] Wojciech Jarosz, Henrik Wann Jensen, and Craig Donner, "Advanced global illumination using photon mapping," *SIGGRAPH 2008 classes,,* August 2008, NY, USA

[9] Henrik Wann Jensen, "Realistic image synthesis using photon mapping," *A. K. Peters, Ltd. Natick,* March, 2001

[10] Henrik Wann Jensen, "Global illumination using photon maps," *Proceedings of the eurographics workshop on Rendering techniques '96,* p.21-30, December 1996, Porto, Portugal

[11] Henrik Wann Jensen, "A practical guide to global illumination using ray tracing and photon mapping," *ACM SIGGRAPH 2004 Course Notes,* p.20-es, August 08-12, 2004, Los Angeles, CA

[12] Henrik Wann Jensen, "Rendering caustics on non-Lambertian surfaces," *Proceedings of the conference on Graphics interface '96,* p.116-121, May 1996, Toronto, Ontario, Canada

[13] Henrik Wann Jensen, "Importance Driven Path Tracing using the Photon Map," *EUROGRAPHICS Workshop on rendering 1995,* p. 359-369, June, 1995.

[14] Irena Notkin, Craig Gotsman, "Parallel Progressive Ray-tracing," *The Eurographics Association 1997,* Volume 16(1997), number 1 pp.43-55

[15] Mark J. Pavicic, "Convenient anti-aliasing filters that minimize bumpy sampling," *Graphics Gems I, edited by Andrew S. Glassner,* pp.144-146, Cambridge, MA: Academic Press, 1990.

[16] Fabio Pellacini, Kiril Vidime, Aaron Lefohn, Alex Mohr, Mark Leone, and John Warren, "Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography," *ACM Transactions on Graphics (TOG),* v.24 n.3, July 2005 1991

[17] Gregory J. Ward, Julie Dorsey , Seth Teller, "The RADIANCE lighting simulation and rendering system," *Proceedings of SIGGRAPH '94,* pp.459-472, New York: ACM Press, July, 1994.

[18] Gregory J. Ward, "Irradiance caching algorithm," *CM SIGGRAPH 2008 classes,* August 2008, NY, USA

[19] Gregory J. Ward,Francis M. Rubinstein, and Robert D. Clear, "A ray tracing solution for diffuse interreflection," *ACM SIGGRAPH 2007 courses,* p.95-92, August 2007, NY, USA

[20] Gregory J. Ward, "Implementation of irradiance caching in radiance," *ACM SIGGRAPH 2008 classes,* August 2008, NY, USA

[21] Matthias Zwicker, "Continuous Reconstruction, Rendering, and Editing of Point-Sampled Surfaces," *PhD Thesis,* ETH Zurich, 2003