International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **6**, December **2018**

 ISSN 1349-4198
pp. 2117–2127

GPU-BASED GLOBAL RENDERING VIA IMPORTANCE-DRIVEN ENVIRONMENT MAP SAMPLING

Jinhua Fu[1,2], Yongzhong Huang[1] and Jie Xu[3,*]

[1]State Key-Laboratory of Mathematical Engineering and Advanced Computing
No. 62, Science Ave., Zhengzhou 450001, P. R. China

[2]School of Computer and Communication Engineering
[3]School of Software
Zhengzhou University of Light Industry
No. 5, Dongfeng Road, Zhengzhou 450002, P. R. China
zzuliteacher@126.com; *Corresponding author: jiexuzz@tom.com

Received April 2018; revised August 2018

Abstract. *In some real-time applications such as 3D games and virtual reality, it has to produce interactive rendering of global illumination for different objects. Global illumination (GI) has very complex lighting condition such as diffuse light, inter-reflection and soft shadows, which is difficult to simulate GI. In this paper, we propose the GPU (Graphics Processing Unit)-based global rendering via importance-driven environment map sampling. We use HDR (High Dynamic Range) environment map to efficiently represent complex global illumination, which is usually capable of GPU implementation. Based on the importance-driven sampling points around bright areas, we obtain fewer but more important illumination points to simulate global illumination. Global rendering equation is deduced and the discretization is acquired using the two-pass algorithm of photon mapping via CUDA (Compute Unified Device Architecture) acceleration. Finally, the results show that we acquire realistic effects of rendering such as inter-reflections, luster, caustics and soft shadows, while keeping an interactive rendering speed.*
Keywords: Interactive rendering, Global illumination, Environment map, GPU implementation

1. **Introduction.** The goal of physically-based global illumination is to render a realistic and physically accurate image of a virtual world. The most relevant light paths from the light sources to the viewer's position have to be traced, simulating light interactions with the transmitting medium, e.g., air or water, and with objects in the scene. Since these interactions are simulated using physically-based illumination models, the resulting image is a correct depiction of what would be seen in a real situation. Although realistic rendering is a very demanding task in the practical applications such as computer graphics, virtual reality and 3D interactive games, it has historically been the domain of high-quality offline rendering due to great computational cost. So the main drawback of physically-based models is that rendering is a time consuming process, thus sampling and fast rendering computation should be researched further.

So far, many methods for realistic rendering of global illumination have been proposed in the past. Some classic rendering methods such as path tracing [1], Monte-Carlo algorithm [2, 3] and radiosity equation [4, 5], can simulate the global illumination efficiently. However, these methods are general in that they can handle arbitrary illumination, geometry and materials in an unbiased manner, they are typically computationally expensive task and may take long to converge, making them infeasible for practical rendering. On

DOI: 10.24507/ijicic.14.06.2117

the other hand, global illumination is the lighting conditions surrounding the rendered object with complex light interactions like area shadowing and diffuse inter-reflection. Instead of a certain number of light sources as used in most current methods, in some practical scenes, the appearance of object is illuminated and influenced by environment light which represents the global illumination. So environment map is effectively used to simulate environment light which contains the light coming from each direction around the object appearance, which has been applied in many special applications, such as Pixar's cartoons [6], translucent materials rendering [7] and mixed reality [8].

In order to accelerate the rendering, some sampling techniques to environment map are presented such as uniform sampling scheme [9] and non-uniform sampling scheme [10]. For rendering view-dependent, glossy surfaces to increase the realism in real-time applications, Luksch et al. [11] introduce a regular sampling scheme for environment maps that relies on an efficient Mipmapping-based filtering step, which minimizes the number of necessary samples for creating a convincing real-time rendering of glossy BRDF (Bidirectional Reflectance Distribution Function) materials. Yang et al. [12] propose a novel method for high dynamic environment sequence sampling to improve rendering quality across environment sequences, which shows decent temporal coherence of their approach in sequence rendering. Similarly, noniterative adaptive sampling [10] is also applied to the IBR (Image-Based Rendering) sampling and reconstruction with a light field setup. However, the method has many errors in RMS (Root Mean Square), especially when the sampling points are growing increase in number. On the other hand, a new visibility sampling technique [13] is presented which shows that efficient all-frequency rendering of dynamic scenes can be achieved by sampling visibility of dynamic objects on the GPU. More recently, radiosity theory is presented to give the rendering equation to implement simulation of global illumination [14, 15]. Although they acquire good effects of global illumination, future improvements for improving the computation still do not consider important illumination distribution such as bright and dark areas.

Nowadays, with the increasing requirement of rendering speed, the main challenge of rendering is not only to approximate the realistic effect, but also develop sufficiently fast method to acquire interactive rendering at higher speed. Motivated by meeting the increasing requirement of interactive applications, our research focuses on sampling of the environment lighting, and realizes the detection of important regions of the rendering scene, such as windows and doors. These parts of the scene often have a small area proportional to that of the entire scene, so paths which pass through them are generated with a low probability. Using efficient evaluation via importance-based sampling methods, and using fast rendering with GPU acceleration [16]: CUDA technique, we obtain more reasonable distribution of sample points to represent important energy information of light source. After the sampling points have been reasonably spread, we can transform the HDR environment map into simplified representation by using the weighted sampling technique. The method proposed in this paper improves sampling efficiency, by taking account of view importance. This method automatically constructs a sampling distribution in locations which are relevant to the bright areas, thereby improving sampling of light paths. Then using two-pass photon mapping, we implement the global illumination algorithm and solve the complex rendering equation approximately. Our method could acquire a physically-based rendering appearance while keeping interactive rendering speed, which is suited to practical applications such as virtual reality and 3D games.

The rest of this paper is organized as follows. The importance-driven environment map sampling for global illumination is given in Section 2. Discretization solution of global rendering is deduced in Section 3. At last, experimental results are discussed in Section 4, and finally conclusions are given in Section 5.

2. Importance-Driven Environment Map Sampling for Global Illumination.

2.1. Environment lighting simulation.

Environment lighting is the lighting conditions surrounding the rendered object, which can represent effects of global illumination. However, environment lighting is difficult to be described and simulated, which is usually incapable of GPU implementation. So an efficient technique to simulate the complex lighting is HDR environment maps such as Grace Cathedral scene [17] shown in Figure 1.

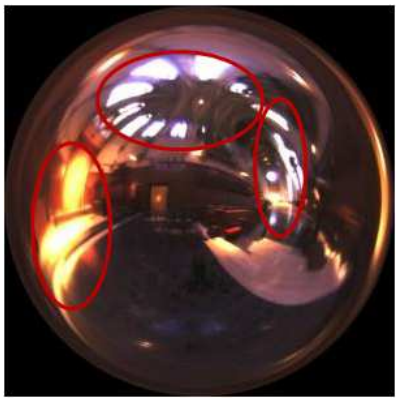


FIGURE 1. One HDR environment map for simulating global illumination

For HDR environment map, there is a great deal of variation in brightness shown as circles in Figure 1. If sampling the map randomly, while it will eventually converge appropriately, the sampling result can be very inefficient. Instead, it is a much better strategy to sample more densely where the map is brightest. This observation is trivial when considering it from a photon casting point of view: physically, brighter regions will cast more photons. So an efficient sampling method for HDR environment map has been implemented as an extension. We make a weighted-average of all estimators where the weights depend on the sampling positions. The basic idea behind Monte Carlo integration is evaluation for some function $f(x)$ over domain Ω as follows

$$I = \int_{\Omega} f(x)dx \tag{1}$$

Also, sometimes it may not be possible to sample a space uniformly. In order to alleviate the problem, we can rewrite the estimator from Equation (1) as

$$I = \int_{\Omega} f(x)dx = \int_{\Omega} \frac{f(x)}{p(x)}p(x)dx \tag{2}$$

where $p(x)$ is a probability density function (PDF) in Ω. We have n different estimators, and the combined estimator is then given by

$$I = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{3}$$

where p_i is the PDF for each estimator, n_i denotes the number of samples from p_i, $X_{i,j}$ are the samples from p_i, for $i, j = 1, \ldots, n$, and all samples are assumed to be independent. w_i is the weighting function which satisfies the following two conditions:

$$\begin{cases} \sum_{i=1}^{n} w_i(x) = 1 \\ w_i(x) = 0 \text{ whenever } p_i(x) = 0 \end{cases} \tag{4}$$

An obvious weighting function would be

$$w_i(x) = c_i \frac{p_i(x)}{q(x)} \tag{5}$$

where

$$q(x) = c_1 p_1(x) + c_k p_k(x) + \cdots + c_i p_i(x) \tag{6}$$

and all coefficients c_i are nonzero and sum to 1.

The shortages of existing methods to render the global illumination are difficult to meet the dual challenges of photo-realistic appearance and rendering speed. There are some aspects of the real world that ray tracing does not handle very well. Perhaps the most important omissions are diffuse inter-reflections and caustics. Also the computation time has to be speeded up greatly to solve the real-time problem in 3D games and virtual reality. So using the efficient sampler in Section 2.2, we can produce a piecewise constant approximation to the HDR environment map and solve the dual challenges. We can sample this map in proportion to some function of its pixel values in the implementation.

2.2. **Weighted importance sampling algorithm.** In general, randomly selecting a set of points for each surface point would be an example of unbiased sampling. This sampling method would be very expensive for real-time rendering applications, and suffers from high variance in the estimate between surface points. Instead, we choose a biased representation of the environment lighting by collapsing the HDR environment map into a set of point lights. These points are then used for all surface points in the rendering. By pre-computing the set of points, this method can be applied to real-time rendering and removes the variance in the integral estimate between adjacent surface points.

In some HDR environment maps such as Grace Cathedral scene [17], there are certain areas that are brighter, where some points can be sampled to represent important energy information of light source using weighted importance sampling. Grace Cathedral scene is a particularly compelling contrast between bright and dark areas. By concentrating sampling to the important brighter areas, we create a better approximation and simplified representation of the illumination integral using fewer sampling points. The weighted importance sampling could be described in Algorithm 1.

Algorithm 1 Weighted Importance Sampling Algorithm

1: Define $\mathbf{A}$ as the monochrome environment map.
2: Suppose $e = \underbrace{(1, 1, \ldots, 1)}_{n}$. Compute a vector $r = \mathbf{A}e$ of row intensities.
3: Define a PDF by normalizing r, and define the corresponding CDF (Cumulative Distribution Function).
4: Similarly, each row is defined as: $A = \left\{ \begin{array}{c} a_1^T \\ \ldots \\ a_m^T \end{array} \right\}$.
5: Define a CDF for each a_i.
6: Generate a set of random tuples in $[0, 1] \times [0, 1]$ for each point.
7: By using binary search or otherwise, find the value i that maps, according to the CDF of r, to the randomly generated value.
8: Given i, find the value j that maps, according to the CDF of a_i, to the other randomly generated value. a_{ij} is the sampled point. Repeat steps 6 to 7 for every random tuple.

From Algorithm 1, the weighted importance sampling is biased towards pixel intensities, and sampled pixels tend to group around important bright areas. For the Grace Cathedral, many important points cluster around the windows and alter in the scene. Therefore, we can get a good approximation where the light is focused at the top hemisphere. We also notice the sparsity of points in the darker areas of the environment map.

To see if our sampling is correct, we plot the sampled points on the Grace Cathedral map as shown in Figure 2. From the result image, we can see most of the samples are located in "bright" places of the map.

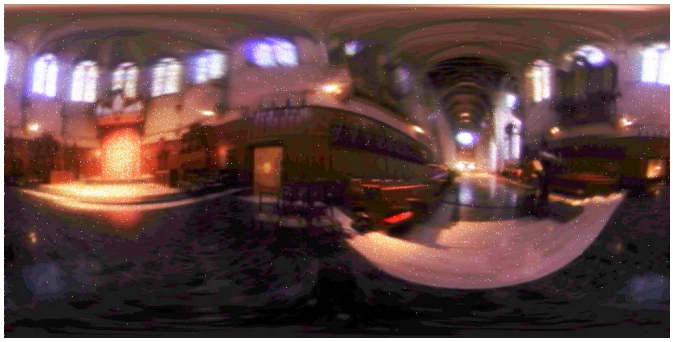


FIGURE 2. Importance sampling on Grace Cathedral map

From Figure 2, because the importance sampling is biased towards pixel intensities, sampled pixels tend to group around bright areas. For the Grace Cathedral, many points cluster around the windows. Therefore, we get a good approximation where the light is focused at the top bright hemisphere.

Computing the total light energy is most naturally performed on a monochrome version of the lighting environment rather than the RGB pixel colors. Such an image can be formed as a weighted average of the color channels of the light probe image, e.g., $Y = 0.2125R + 0.7154G + 0.0721B$ following ITU-R Recommendation BT.709 color space. The relative darkness of the rest of the Cathedral forces most light sources to be situated at the upper half of the spherical environment map texture. Because we lack light sources with $\phi < 0$, any pixel with a normal pointing towards the floor will have a poor approximation. The pseudocode of importance sampling is as follows:

```
function result=ImportanceSampling(img, fileinfo, n)
monocrome=img(:,:,1)*0.2125 + img(:,:,2)*0.7154 + img(:,:,3)*0.0721;
N=fileinfo.width;
M=fileinfo.height;
x=1:N;
y=1:M;
phi=linspace(-pi/2,pi/2, M);
rowPDF=sum(monocrome, 2)'.*cos(phi);
rowPDF=rowPDF./sum(rowPDF);
rowCDF=cumsum(rowPDF);
colPDF=monocrome./repmat(sum(monocrome,2),1,N);
colCDF=cumsum(colPDF,2);
result=img;%zeros(M,N);
for(i=1:n)
c1=min(find(rand<rowCDF));
c2=min(find(rand<colCDF(c1,:)));
result(max(c1-3,1):min(c1+3,M),max(c2-3,1):min(c2+3,N),:)=0;
result(max(c1-2,1):min(c1+2,M),max(c2-2,1):min(c2+2,N),:)=255;
end%end for
end%end function
```

From the pseudocode of importance sampling above, Line 2 indicates that, we obtain a monochrome version of the lighting environment via weighted average of the color channels in ITU-R Recommendation BT.709 color space. *phi* in Line 7 refers to the angle subtended by each pixel determined by the resolution of the environment map. From Line 8 to Line 10, we compute the sampling value of PDF in each row of environment map. From Line 11 to Line 12, we evaluate the sampling value of CDF according to the PDF in each column of environment map. From Line 15 to Line 16, we obtain two coefficients to evaluate the weight in Equation (5). From Line 17 to Line 18, we finally get result of importance sampling where the light is focused at the top hemisphere.

3. **Discretization Solution of Global Rendering.** Photon mapping is a two-pass algorithm of global illumination that approximately solves the complex rendering equation, which is used to calculate the global illumination effects such as indirect lighting and caustics. The first pass builds the *photon map* by emitting photons from the light sources into the scene and storing them in a *photon map* when they hit non-specular objects. The second pass, the rendering pass, uses statistical techniques on the *photon map* to extract information about incoming flux and reflected radiance at any point in the scene.

The *photon map* can be seen as a representation of the incoming flux. To compute radiance we need to integrate this information. This can be done using the expression for reflected radiance:

$$L_r(x, \omega_o) = \int_{\Omega_x} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) |N_x \cdot \omega_i| d\omega_i \tag{7}$$

where L_r is the reflected radiance at point x in direction ω_o. Ω_x is the hemisphere of incoming directions. f_r is the BRDF at point x and L_i is the incoming radiance in incoming direction ω_i. $|N_x \cdot \omega_i|$ is the dot product between normal N_x and incoming direction ω_i.

In order to evaluate the rendering Equation (7) we have to acquire information about the incoming radiance. Since the *photon map* provides information about the incoming flux Φ_i, we need to rewrite this term. This can be done using the relationship between radiance L_i and flux Φ_i:

$$L_i(x, \omega_i) = \frac{d^2\Phi_i(x, \omega_i)}{\cos\theta_i d\omega_i dA_i} \tag{8}$$

where dA_i is a micro-facet area around x. $\cos\theta_i = |N_x \cdot \omega_i|$.

Then we can deduce the integral based on Equation (7) and Equation (8):

$$L_r(x, \omega_o) = \int_{\Omega_x} f_r(x, \omega_i, \omega_o) \frac{d^2\Phi_i(x, \omega_i)}{\cos\theta_i d\omega_i dA_i} |N_x \cdot \omega_i| d\omega_i = \int_{\Omega_x} f_r(x, \omega_i, \omega_o) \frac{d^2\Phi_i(x, \omega_i)}{dA_i} \tag{9}$$

The incoming flux Φ_i is approximated using the *photon map* by locating the n photons that has the shortest distance to x. Each photon p has the power $\Delta\Phi_p(\omega_p)$ by assuming that the photons intersect the surface at point x we obtain

$$L_r(x, \omega_o) \approx \sum_{p=1}^{N} f_r(x, \omega_p, \omega_o) \frac{\Delta\Phi_p(x, \omega_p)}{\Delta A} \tag{10}$$

The procedure can be imagined as expanding a sphere around x until it contains n photons (see Figure 3) and then using these n photons to estimate the radiance.

Equation (10) still contains Δ which is related to the density of the photons around x. By assuming that the surface is locally flat around x we can compute this area by

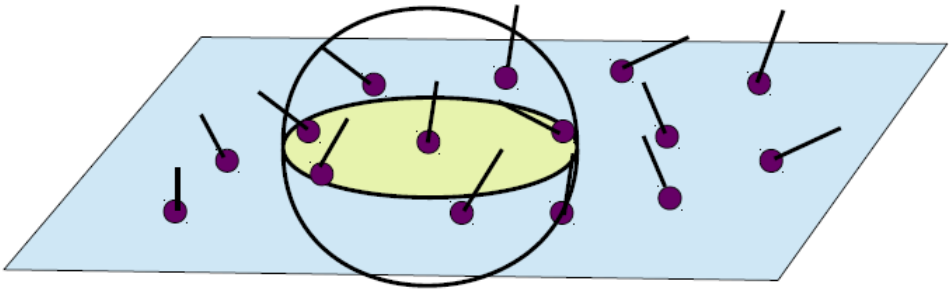


FIGURE 3. Radiance estimation via the nearest photons in the *photon map*

projecting the sphere onto the surface and use the area of the resulting circle. This is indicated by the hatched area in Figure 3 and equals:

$$\Delta A = \pi r^2 \tag{11}$$

where r is the radius of the sphere $-ie$, which is the largest distance between x and each of the photons.

This results in the following equation for computing reflected radiance at a surface using the *photon map*:

$$L_r(x, \omega_o) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x, \omega_p, \omega_o) \Delta\Phi_p(x, \omega_p) \tag{12}$$

From Equation (12), this estimate is based on many assumptions and the accuracy depends on the number of photons used in the *photon map* and in the formula. As more photons are used in the estimate and in the *photon map*, Equation (12) becomes more accurate.

Then we describe the mapping of the algorithm described above, to DirectX 11 compatible GPU. A summary of this implementation is given in the steps below.

1) Render sampling results of the HDR environment map each of the $n_x \times n_y \times n_z$ grid points: We use floating point off-screen render targets to render and to store the HDR maps.

2) Compute M coefficients of weighted importance sampling to represent radiance captured at the grid points: We compute the coefficients in pixel shaders using multiple passes. These coefficients are stored as M, 3D texture maps of dimension $n_x \times n_y \times n_z$.

3) Compute L_r: We use a pixel shader (i) to tri-linearly interpolate the radiance coefficients f_r from the 3D textures in GPU hardware based on the 3D texture coordinates of the surface point visible through the pixel, (ii) to compute $\Delta\Phi_p$, the central direction specific coefficients, and finally (iii) to compute indirect light from the $\Delta\Phi_p$ and the interpolated f_r.

4) Repeat steps 1)-3) for a number of times as the time budget allows.

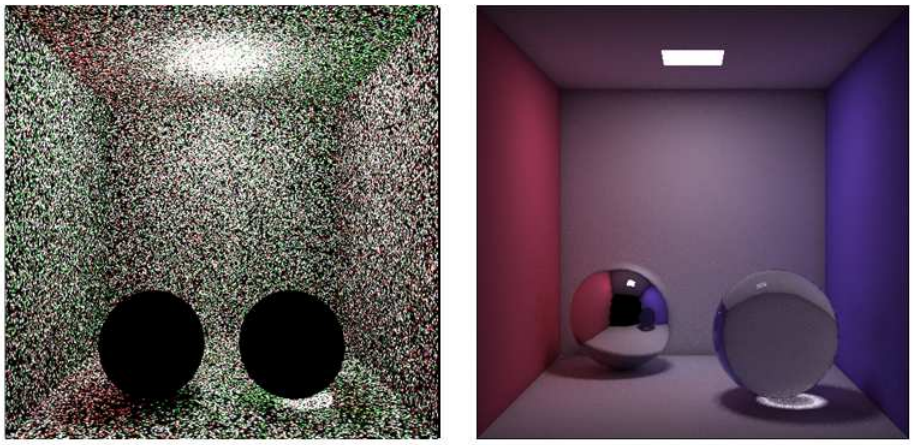
5) Render image with lighting from global illumination: We compute direct light from local light model with shadow support. We add this direct light to indirect light computed from step 4), and assign the result to the image pixel.

All the steps of our algorithm are implemented using fragment/vertex shaders. We have implemented the algorithm in DirectX 9. Note that the standard OpenGL library (Version 2.6) does not support many of the DirectX 9 features. We have made use of ARB (Architecture Review Board) and vendor specific extensions to make use of such features. The OpenGL specification is overseen by the ARB, which consists of a set of companies in creating a consistent and widely available API. The OpenGL standard allows individual vendors to provide extended functionality through extensions as new

technology is created. An extension is then distributed in two parts: as a header file which contains the extension's function prototypes, and as the vendor's device driver. Each vendor has an alphabetic abbreviation that is used in naming their new functions and constants. For example, Nvidia's abbreviation is used in defining their proprietary function *glCombinerParameterfvNV*() and their constant *GL_NORMAL_MAP_NV*. It may happen that more than one vendor agrees to implement the same extended functionality.

4. **Results.** We have implemented an efficient rendering based on sampling and discretization of global illumination via CUDA acceleration. With Intel(R) Core(TM) i7-5500U CPU @ 2.40GHZ and NVIDIA Geforce GTX 590, we have realized the method using CUDA acceleration and C++ programming in VS2016. We use the 1024∗512 Grace Cathedral environment map from Paul Devebec's Light Probe [17], shown in Figure 2, with the Stanford Happy Buddha model (containing more than 1 million polygons).

Because of the sparsity of points in the importance sampled results, we acquire the simplified representation of the illumination integral using fewer sampling points. By evaluating the discretization of the two-pass photon mapping to the GPU, we have achieved rendering results of global illumination at approximately 49 FPS (Frames Per Second), at the cost of providing a rough approximation of the solution. We firstly cast some photons in the Cornell box containing glass and chrome spheres. Then we sample photons according to brightness of environment map using weighted importance sampling and render the scene with global illumination as shown in Figure 4. We can obtain global illumination such as soft shadows, caustics and inter-reflection.



(a) (b)

FIGURE 4. "Cornell box" with glass and chrome spheres: (a) photons in the corresponding photon map and (b) rendering with global illumination

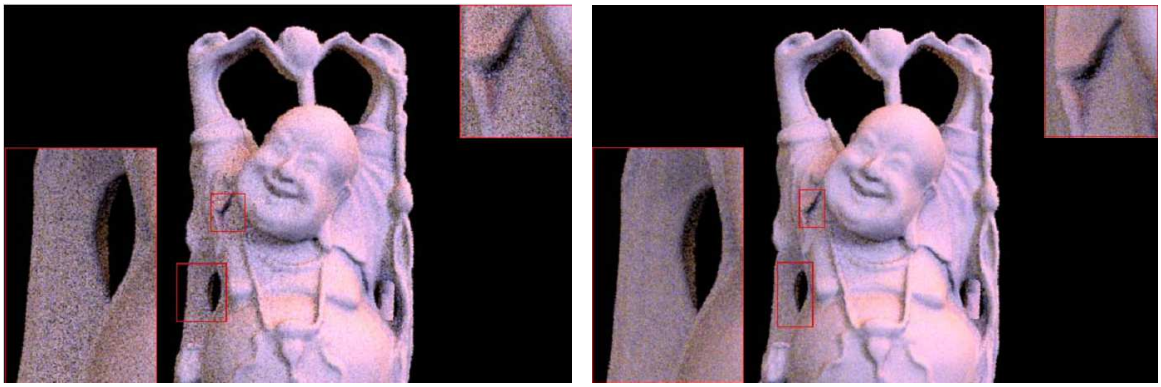
In order to highlight physically-based global illumination, we render the teapot model and two glass spheres in HDR Grace Cathedral environment as shown in Figure 5 and Figure 6, and focus on rendering effects of global illumination.

From Figure 5 and Figure 6, using our sampling we observe that rendering in the Grace Cathedral environment produces a physically-based appearance such as refractions, inter-reflection, soft shadows, luster and caustics, which is in accord with physical phenomena. Hence, good quality of rendering can be enhanced and sense of reality can be acquired via our method.

Then we implement rendering on the Stanford Happy Buddha model to compare our method with classic uniform sampling [9]. The Buddha model has complex scene geometry and surface orientations. We use low-discrepancy sequences to produce random samples

FIGURE 5. One global rendering scene

FIGURE 6. Another global rendering scene



(a) Sampling via uniform sampling

(b) Sampling via our method

FIGURE 7. Comparison of denoising effect on Buddha model in Grace Cathedral light environment

in sampling. Results of rendering Buddha with 16 samples per shading point are shown in Figure 7.

From Figure 7, uniform sampling suffers from the sample-clumping problem and introduces significant noise at surface points where bright light sources are occluded by the geometry of the model (marked by red rectangles). Our method, with its better distribution of the samples, reduces the error in such regions. On the other hand, for a light sample, averagely speaking, uniform sampling needs to be computed on the whole map, while our method samples only on a reasonable and important local region.

Later we compare rendering results of three sampling techniques. Figure 8 shows the rendering results using classic uniform sampling, adaptive sampling [10] and our sampling method respectively.

From Figure 8, uniform sampling on each region introduces significant noise for such high glossy surfaces. Adaptive sampling with 32 samples gives nice static rendering result, but fails to catch part of the high light, which is observed on the hands and the arms of the Buddha. However, our method can greatly reduce the noise and catches more hight light at 32 final samples per shading point, since the surface orientation and the glossy BRDF component are both considered for sample selection. So we have found that

(a) Sampling via uniform sampling

(b) Sampling via adaptive sampling

(c) Sampling via our method

FIGURE 8. Different sampling results on Buddha model in Grace Cathedra scene

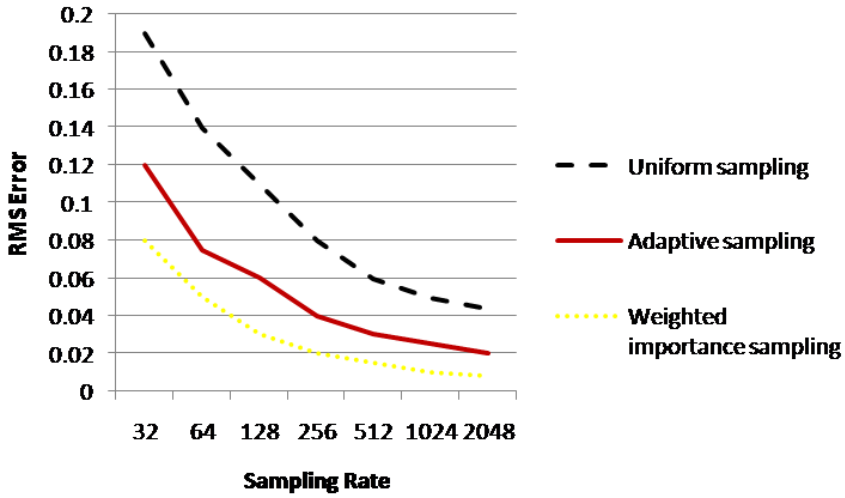


FIGURE 9. RMS error for the rendering scene in Grace cathedral with three methods

our method is enough to produce good renderings in surface fitting degree and surface flatness. This shows that with our technique, it is possible to render glossy model under complex lighting conditions with fewer number of samples.

Finally, Figure 9 compares the RMS error of three sampling techniques. The reference image is rendered by standard importance sampling with 10,000 samples per shading point. Weighted importance sampling presents the lower RMS error at all sampling rates over classic uniform sampling [9] and adaptive sampling [10]. This improvement is more effective at lower sampling rates.

Above all, compared with the existing biased lighting techniques, the presented scheme produces unbiased rendering results with less noise, less RMS error, more realistic appearance and better shadow boundaries, particularly at low sampling rates. Our proposed method can give a reasonable distribution of the samples and also helps to overcome the sample-clumping problem in traditional illumination-based sampling method.

5. **Conclusions.** We have implemented the global rendering based on weighted importance sampling and two-pass photon mapping. In this paper, we can give the simplified

representation of HDR via weighted importance sampling and realize a lower RMS error at all sampling rates. Using two-pass photon mapping, we can simulate the global illumination efficiently. Effects of global illumination such as inter-reflections, soft shadows, caustics and luster can be acquired, which is in accord with physical phenomena and can enhance the realism of rendering appearance greatly.

In many practical applications such as 3D video games and online applications, rendering of global illumination still cannot meet fast interactive requirements. so how to speed up rendering via GPU is an interesting exploration later.

Acknowledgment. This work was supported by the National Natural Science Foundation of China under Grant (No. 61672470), the Intergovernmental Special Program of National Keypoint Research Project (Nos. 2016YFE0100300, 2016YFE0100600), and the Science and Technology Foundation of Henan Province Education Department of China (No. 14A520061).

REFERENCES

[1] E. P. Lafortune and Y. D. Willems, Rendering participating media with bidirectional path tracing, *Eurographics Workshop on Rendering Techniques*, pp.91-100, 1996.

[2] L. Szirmay-Kalos, Monte-Carlo global illumination methods – State of the art and new developments, *Spring Conference on Computer Graphics*, 1999.

[3] Y. C. Lai, S. Chenney, S. Chenney et al., Photorealistic image rendering with Population Monte Carlo energy redistribution, *Proc. of the 18th Eurographics conference on Rendering Techniques*, pp.287-295, 2007.

[4] A. R. Naghsh-Nilchi and S. Daroee, Iterative sinc-convolution method for solving radiosity equation in computer graphics, *Electronic Trans. Numerical Analysis*, vol.23, no.1, pp.251-262, 2006.

[5] Y. Dong, S. Lin and B. Guo, Modeling and rendering subsurface scattering using diffusion equations, in *Material Appearance Modeling: A Data-Coherent Approach*, Springer Berlin Heidelberg, 2013.

[6] D. Garcia, I. Quilez, D. Dixon et al., Environment rendering optimization for Pixar's the good dinosaur, *ACM SIGGRAPH*, 2015.

[7] G. Li and Y. Guo, GPU-based rendering using discrete diffusion model and cube environment mapping, *ICIC Express Letters, Part B: Applications*, vol.7, no.4, pp.805-811, 2016.

[8] F. Zhang, Y. Zhao, Z. Wang et al., An efficient all-frequency environment rendering method for mixed reality, *IEEE International Congress on Image and Signal Processing, Biomedical Engineering and Informatics*, pp.358-362, 2017.

[9] P. M. Lam, C. S. Leung, T. T. Wong et al., Uniformly sampling multi-resolution analysis for image-based relighting, *Journal of Visual Communication and Image Representation*, vol.21, no.7, pp.693-706, 2010.

[10] H. Wang, Z. Ai, Y. Cao et al., A parallel preintegration volume rendering algorithm based on adaptive sampling, *Journal of Visualization*, vol.19, no.3, pp.437-446, 2016.

[11] C. Luksch, R. F. Tobler and M. Wimmer, Real-time rendering of glossy materials with regular sampling, *Visual Computer*, vol.30, no.6, pp.717-727, 2014.

[12] Y. Yang, C. Yu, L. Wan et al., High dynamic environment sequence sampling, *ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pp.123-131, 2014.

[13] R. Wang, M. Pan, X. Han, W. Chen and H. Bao, Parallel and adaptive visibility sampling for rendering dynamic scenes with spatially varying reflectance, *Computers and Graphics*, vol.38, no.1, pp.374-381, 2014.

[14] W. Song, J. Fu and Y. Zhang, GPU-based rendering of global illumination using discretization of radiosity equation, *ICIC Express Letters*, vol.10, no.4, pp.943-948, 2016.

[15] P. Hedman, T. Karras and J. Lehtinen, Sequential Monte Carlo instant radiosity, *IEEE Trans. Visualization and Computer Graphics*, vol.23, no.5, pp.1442-1453, 2017.

[16] H. T. Chang, H. W. Peng and C. H. Tsai, CUDA-accelerated rendering of fireworks in nearly ultra high definition videos, *IEEE the 2nd International Conference on Multimedia Big Data*, pp.251-254, 2016.

[17] *Light Probe Image Gallery*, http://www.pauldebevec.com/Probes/.