國立交通大學

多媒體工程研究所

碩 士 論 文

補償光子在漸進性光子映射中之應用

Progressive Photon Mapping With Compensation Photons

研 究 生：張晉豪

指導教授：施仁忠 教授

蔡侑庭 教授

中 華 民 國 一百零一 年 八 月

補償光子在漸進性光子映射中之應用

Progressive Photon Mapping With Compensation photons

研 究 生：張晉豪　　Student：Jing-Hao Chang

指導教授：施仁忠　　Advisor：Prof. Zen-Chung Shih

蔡侑庭　　Prof. Yu-Ting Tsai

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Multimedia Engineering

August 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年八月

補償光子在漸進性光子映射中之應用

研究生: 張晉豪　　指導教授: 施仁忠教授

蔡侑庭教授

國立交通大學多媒體工程研究所

摘要

漸進性光子映射演算法(Progressive Photon Mapping)由傳統全域照明技術之一的光子映射演算法(Photon Mapping)所改進而來，改變了傳統光子映射演算法的架構，使得光子只需儲存擊點(hit point)的資訊。

本篇論文根據光子映射演算法提出一個易實作且直覺的補償光子(Compensation Photon)應用在漸進性光子映射演算法。在互動式的場景中，當場景中物體位置變化時藉由補償光子可以不用每個畫面都須重新設置擊點的位置，並且重新潑灑所有光子。補償光子將移動的物體影響至場景中的能量抵銷，包含了物體移動範圍內的能量以及物體影響至場景的能量，使得下一個畫面中不用重新潑灑所有光子，只需潑灑物體移動的範圍以及影響到的區域所需的光子即可。不用計算與前一個畫面相同能量的區域，因此可以使得漸進性光子映射演算法在互動式場景中達到加速的效果。

Progressive Photon Mapping With Compensation Photons

Student: Jing-Hao Chang Advisor: Prof. Zen-Chung Shih

Prof. Yu-Ting Tsai

Institute of Multimedia Engineering
National Chiao-Tung University

ABSTRACT

Progressive photon mapping is a simple and robust progressive global illumination algorithm based on photon mapping. It changes traditional photon mapping's framework , such that we only need to store the hit point information of each photon.

In this thesis, we introduce a compensation photon to the progressive photon mapping, which is simple and easy to implement. In interactive environments, when an object moves, via the compensation photon, there is no need to set all hit point and re-emit all the photons. Using compensation photon to offset the power in the scene coming from the moving object, we can find the areas there have the same power as the previous frame. There is no need to re-emit all the photons to render the entire frame. We just emit to the areas where compensation photon has offset. Therefore we save the time which is taken by emitting unnecessary photons to render the scene.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisors, Prof. Zen-Chung Shih and Prof. Yu-Ting Tsai for their guidance and patience. Without their encouragement, I would not complete this thesis. Thanks also to all the members in Computer Graphics and Virtual Reality Laboratory for their reinforcement and suggestion. I want to thank to all the people who ever supported me during these days. Finally, I will sincerely dedicate this thesis to my parents.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

In computer graphics, real-time global illumination in dynamic scene is still a big challenge today. Due to the inherent complexity of light transport, causing effects such as interreflection, caustics, many global illumination approaches are built on ray tracing[2] or photon mapping[7]. Conventional rendering algorithms can render high-quality images but take so much time. With the enhancement of GPU architectures in recent years, conventional rendering algorithms can improve performance by using parallel computing to obtain high quality image.

Progressive photon mapping[6] and related researches have been proposed in recent years. Progressive photon mapping improves the solution of the rendering equation by including specular-diffuse-specular (SDS) light paths. However, computing a single noise-free image is still time-consuming. Especially for animations, the computation time grows linearly with the number of frames if a consistent simulation result is required. Up to now, there is no method to speed up progressive photon mapping in the dynamic scene especially for the object moving without predefined path.

To handle global illumination in dynamic scenes, alleviating the needs for explicit visibility computation is very important. Antiradiance[4] proposed to treat visibility implicitly, and compensate for light transmitted extraneously. Based on this idea, we propose the

compensation photon to compensate extraneous light transmission. Like antiradiance, our approach shifts visibility computation to local iteration by emitting additional directional compensation photon in the scene. In animation, there is no need to emit all the photons to render the next frame.

The goal of this thesis is to speed up the progressive photon mapping in dynamic scenes. In our proposed approach, we add compensation photons in the progressive photon mapping architecture to compensate the energy. Our major contributions are as follows:

1. We propose a new type of photon, called the compensation photon.
2. A reformulation of progressive photon mapping architecture for dynamic scenes.
3. Speed up the performance of the original progressive photon mapping in dynamic scenes.

1.2 System Overview

An overview of our proposed rendering architecture is shown in Figure1.1. Our method is built based on the progressive photon mapping. Before the objects move, we need to eliminate the different illumination between frames. In order to achieve this purpose, we emit the compensation photons and update the information of hit points. After tracing any types of photon, we gather the photons to evaluate radiance and update the information of hit points in the gathering pass. After the objects move, we need to compensate the illumination when computing the current frame. We emit the positive radiance photons along two types of path. We will discuss this in later chapters. After emitting enough photons, then we can get the image of current frame.

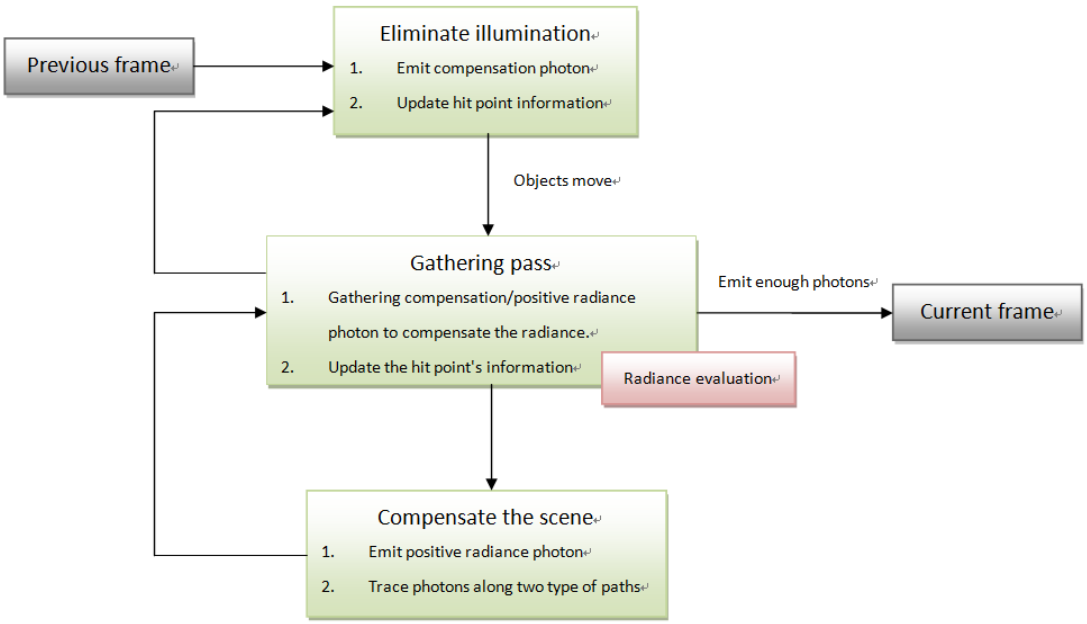


Figure 1.1 The flowchart of our proposed rendering architecture.

Chapter 2
Related Works

In this chapter, we review previous related works. First we focus on the photon mapping and progressive photon mapping. Then we briefly survey real-time global illumination algorithms for the dynamic scene.

2.1 Photon Mapping

Photon Mapping[8] is a two-pass method to solve the rendering equation. In the first step, photons are emitted from the light sources, where each photon stores an equal portion of the total emitted flux. Using Russian Roulette to distribute photons along the scene and store the photons which will be on the diffuse surfaces finally. In the second step, the algorithm determines the radiance for each pixel by accumulating the flux of all the photons that are found inside a search area around the pixel position. To reduce the cost of final gathering step, Ma and McCool[11] use a hash grid rather than a k-d tree to store the photons. Photon Mapping is a biased technique that effectively blurs the real radiance value due to the fixed search radius. But it can render the caustic effect effectively.

2.2 Progressive Photon Mapping

Photon Mapping distributes a finite number of photons inside a fixed search radius. To remove this limitation, Progressive Photon Mapping[6] extends this idea to progressive photon statistics. Progressive Photon Mapping changes the original architecture of the photon mapping. Instead of storing the original photons, only the number and the total flux inside the

search region are stored. Unlike the photon mapping, Progressive Photon Mapping is a multi-pass algorithm, as shown in Figure 2.1. The first pass is the ray tracing pass. An ordinary ray tracing is used to find all the visible surfaces in the scene visible through each pixel. For each ray path, the algorithm stores all hit points along the path where the surface has a non-specular component in the BRDF. Then the subsequent passes use the photon tracing. In the photon tracing step, it accumulates photon power at the hit points found in the ray tracing pass. At each photon tracing pass, it emits a given number of photons into the scene and traces them to build a photon map. After each photon tracing pass, it loops through all hit points and finds photons within the radius of each hit point. Finally they use the added photons to refine the estimate of the illumination. The radiance at each hit point is evaluated as follows :

$$\widehat{\mathrm{N}}(\mathrm{x}) = \mathrm{N}(\mathrm{x}) + \alpha\mathrm{M}(\mathrm{x}) \tag{1}$$

$$\widehat{\mathrm{R}}(\mathrm{x}) = \mathrm{R}(\mathrm{x}) - d\mathrm{R}(\mathrm{x}) = \mathrm{R}(\mathrm{x})\sqrt{\frac{\mathrm{N}(\mathrm{x})+\alpha\mathrm{M}(\mathrm{x})}{\mathrm{N}(\mathrm{x})+\mathrm{M}(\mathrm{x})}} \tag{2}$$

$$\tau_{\widehat{\mathrm{N}}}(x, \vec{\omega}) = \tau_{\mathrm{N}+\mathrm{M}}(x, \vec{\omega})\frac{\mathrm{N}(\mathrm{x})+\alpha\mathrm{M}(\mathrm{x})}{\mathrm{N}(\mathrm{x})+\mathrm{M}(\mathrm{x})} \tag{3}$$

$$L(x, \vec{\omega}) = \frac{1}{\pi\mathrm{R}(\mathrm{x})^2}\frac{\tau(x,\vec{\omega})}{N_{emitted}} \tag{4}$$

where N is the number of photons detected inside a circular region with radius R in current photon tracing pass. M is the number of photons found in next photon tracing pass, but only a fraction $\alpha, 0 < \alpha < 1$, of new photons are needed. While the number of accumulated photons is increasing, the radius R should be reduced. Accordingly, the flux τ is also adjusted. After each photon tracing pass, the radiance at the each hit point can be evaluated. Then the photon tracing pass is updated until

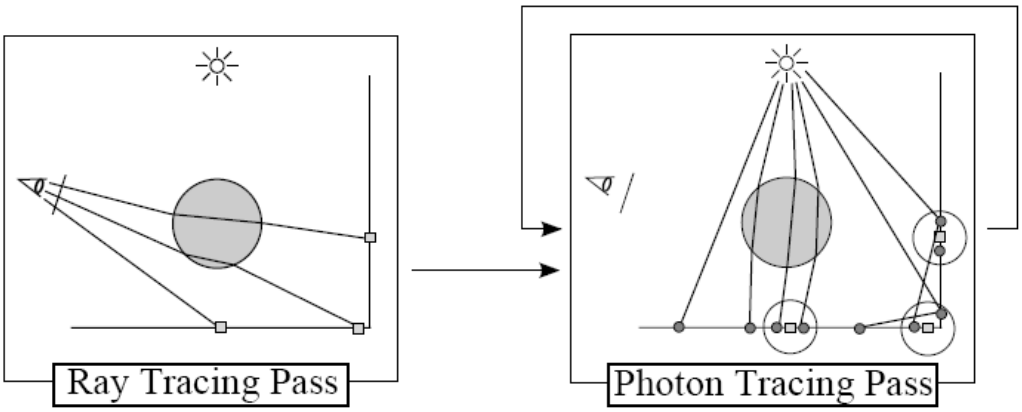


Figure 2.1 Progressive photon mapping uses ray tracing in the first pass followed by one or more photon tracing passes.

2.3 Instant Radiosity

Instant radiosity[9] is a widely used algorithm for approximating global illumination in interactive rendering. Instant radiosity is a bidirectional method that first creates a set of virtual point lights(VPLs) using random walks that trace light paths starting from the primary light sources. It is similar to the photon tracing. Direct contribution of these VPLs then approximates the entire multi-bounce light transport in the scene. The conceptually simple idea is decomposing global illumination to point light source, and combined with shadow mapping. Due to the shadow computation has significant impact on the rendering performance, there are a lot of works tried to reduce this cost. For example, the reflective shadow map[3] treats each viewing pixel from light source as a VPL. Then it computes the contribution of each surface normal while ignores the occlusion of objects. The imperfect shadow map[15] bases on the observation that shadows caused by indirect lighting are more blurred. So it renders low resolution shadow map from each VPL, and then use pull-push to fill holes of coarse shadow map. It uses these imperfect shadow maps to solve the occlusion problem of traditional VPL.

But the major limitation of instant radiosity is that it is only suitable for the scene with diffuse or moderately glossy surfaces and only support once bouncing itself.

Figure 2.2 The results of Instant Radiosity

2.4 Global Illumination in Dynamic Scenes

In Stochastic Progressive Photon Mapping[5], Weiss and Grosch[20] proposed methods to reduce the cost of the computation in dynamic scene like objects moving on a pre-defined path, and material/texture changes. They first load both the static scene and the dynamic geometry for all K frames into main memory. By re-using photon and hit point information for multiple frames, the computation time can be reduced. But there is a big limitation that it can only compute the objects on pre-defined path.

Dachsbacher[4] proposed the antiradiance to reduce the computation time of explicit visibility. They observe that if visibility is ignored, some light is transmitted extraneously through opaque objects and needs to be canceled out. This is why they introduce a new quantity called antiradiance which corresponds to the light that needs to be removed. It shifts visibility computation to simple local iterations by maintaining additional directional

antiradiance information with samples in the scene which is easy to parallelize on a GPU. Our proposed method use a similar idea to create the compensation photon which reduces the radiance which exists in the scene at the previous frame and does not exist at the current frame.

Chapter 3
Algorithm

In this chapter, we discuss our proposed algorithm to speed up the progressive photon mapping. We render direct illumination by using a similar phong lighting model based on the demo of progressive photon mapping of NVIDIA Optix. We only implement our algorithm on indirect illumination and rigid body object. The following figures in this chapter are shown as the indirect illumination part of the scene. All the processes are performed on the GPU. Before the objects move, we perform the following steps for a static frame:

1. Ray tracing pass.
2. Photon tracing pass.
3. Gathering pass.
4. Iterate photon tracing and gathering pass until enough photons emitted.

When we start to move the objects, we perform:

1. We emit the compensation photons towards the area where the moving objects may affect the illumination in the scene.
2. Gathering the compensation photons to render the scene.

After we move the objects, for the dynamic frames, we perform:

1. Update the information of hit points.
2. We emit the positive radiance photons to compensate the indirect radiance that the next frame needs.
3. Gathering the positive radiance photons to render the scene.
4. Repeat step 6 until enough photons emitted.

3.1 Overview

In this section, we describe our concept of our algorithm. In progressive photon mapping, if objects are moved in the scene, it would do the algorithm again and do not keep any information of previous frame. The concept of our algorithm is to keep the information of previous frame and use the information to render the next frame effectively. Most important information in the scene is the radiance. We only discuss indirect radiance in our algorithm. How we keep the indirect radiance is to eliminate the different indirect radiance between two frames.

In figure 1.1, we introduced overview of our algorithm. The key points in our algorithm are eliminating illumination and compensating the scene. We describe the process of these two parts more detail here. To eliminate the indirect radiance, we divide the indirect radiance into two parts. The first part is the indirect radiance bounced by moving volume. Before the objects are moved, we emit compensation photon towards the moving volume to bounce the photons to eliminate the indirect radiance. So we first define the emitting region of compensation photons, then we emit the photons within this region. We discuss more detail in section 3.3. After each compensation photon tracing pass, we calculate the radiance in gathering pass. The other part is the indirect radiance bounced to the moving volume of moving objects. We directly update the hit points in the moving volume to set the radiance to zero. The flow chart of eliminating radiance is shown as figure 3.1. We update the hit points after objects are moved.

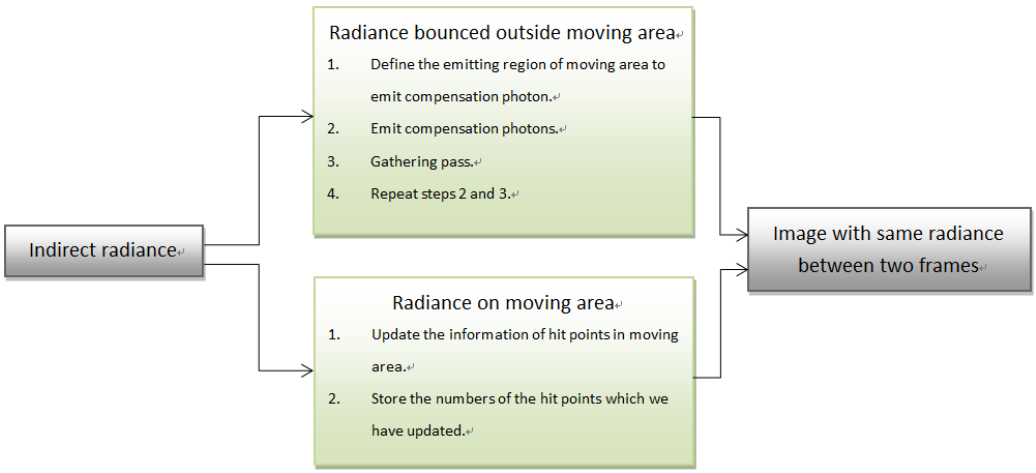


Figure 3.1 The flow chart of eliminating radiance.

To compensate the radiance the next frame needs, we also compensate two parts of indirect radiance. To compensate the radiance on moving volume, we use testing photons to get the directions. We use these directions to emit the positive radiance photons to bounce the photons to the moving volume. We discuss more detail in section 3.5. To compensate the indirect radiance outside the moving volume, we emit the positive radiance photons towards the moving volume by using the same emitting region of compensation photons. The flow chart of compensating the scene is shown as figure 3.2.

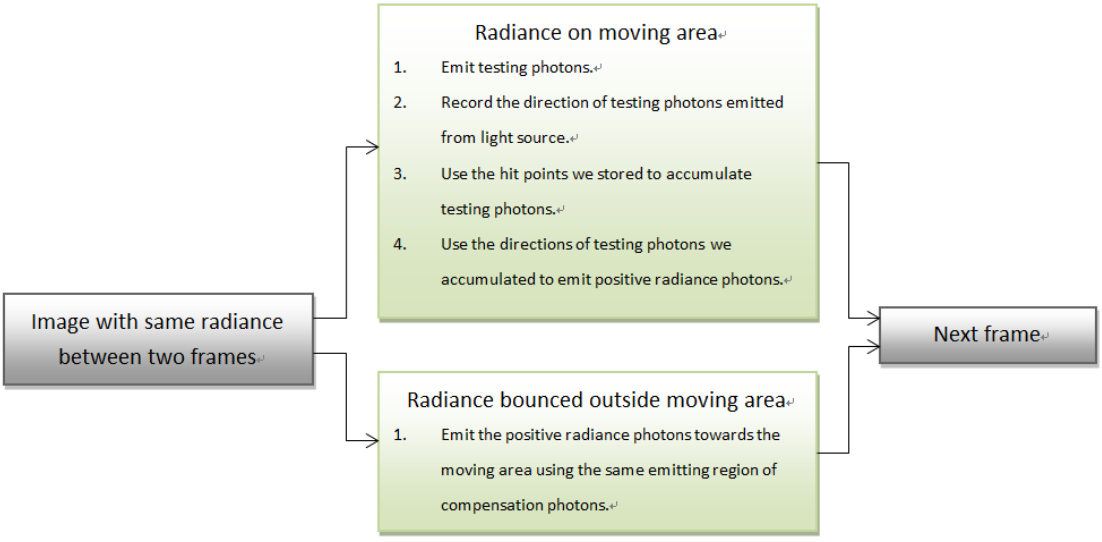


Figure 3.2 The flow chart of compensating the scene.

3.2 Data Structures

Similar to the progressive photon mapping, for each hit point, we store its coordinate and the normal vector, the ray direction, scaling factors including BRDF, the associated pixel location, a radius, the intercepted flux, and the number of photons within the radius.

After the ray tracing pass, we get the information of each hit point. Then we trace the photons along the scene. At each hit point, we obtain the energy from light source. In order to attach photons on object surface, we build an one-dimensional buffer called ***photon map***. The size of photon map depends on the number of the photons emitted in each pass. All of the ray tracing, photon tracing, and gathering passes are performed on GPU by using Nvidia Optix[13].

We use a list to store the serial number of the hit points which we update. We discuss this in section 3.4. We store the directions of testing photons emitted from light source. We discuss this in section 3.5.

3.3 Compensation Photon

In the progressive photon mapping, if there is an object moved in the scene, the algorithm will do the ray tracing pass again and emit all the photons. We found that, when the objects moved in the scene, the radiance in the scene may not change at every frame. So we need to detect and keep the same radiance between the current frame and the next frame.

We propose a new type of photon called the ***compensation photon***. We use this photon to eliminate the indirect radiance in the scene that is not needed for the next frame. Then we can get a frame, which only contains the same radiance between two frames. We observe that

the most difference between the current frame and the next frame is the indirect radiance bounced to the moving volume of the moving objects, as the yellow area shown in Figure 3.3. We set the size and the position of the moving volume in ray tracing pass. We compare the positions of hit points in the current frame with the positions of hit points in the next frame. The area where the positions of the hit points are different is called moving volume. Furthermore, the indirect radiance produced by photons hitting on the moving volume and bounced to side is also different. The major function of the compensation photon is to eliminate the indirect radiance produced by the moving volume.

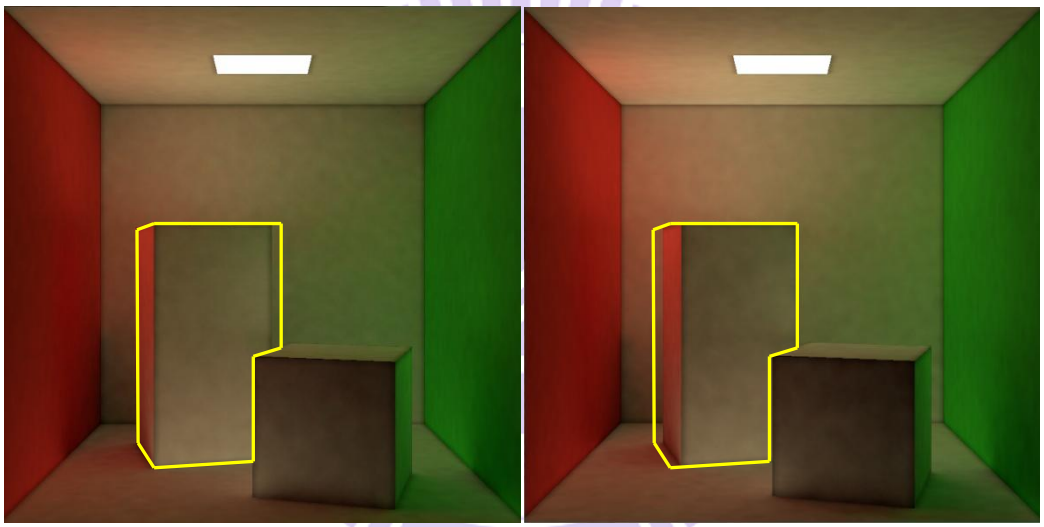


Figure 3.3 Two images only render indirect illumination. (a) Before tall box moves. (b) After tall box moves.

To eliminate the indirect radiance outside the moving volume, we should emit the compensation photons towards the moving volume. The compensation photons hit the moving volume and bounced to side to eliminate the indirect radiance. Before we emit the compensation photons, we store the vertices of the object moved before and after. We use a point light to emit the compensation photons. If the light source of the scene is area light, we put the point light on the central of the area light. We should limit the emitting region of the

point light. To define the region where we should emit the photons, first we set the direction of the point light by connecting the central of the vertices with the point light. And then we project the vertices on the plane. The plane's normal is the direction of point light. To set the position of plane, we set the plane through the central of the vertices. To project the vertices, we connect the vertices with point light and use the connection to calculate the intersection with the plane. We use a bounding rectangular to enclose all the vertices on the plane as moving area of moving objects as shown in Figure 3.4. Finally, we randomly emit the compensation photons through this bounding rectangular to eliminate the indirect radiance. In each compensation photon pass, we emit 256x256 photons. In Figure 3.3, the radiance of yellow areas is eliminated by the compensation photons. Note that we trace the compensation photons before the objects moved, because we need to eliminate the indirect radiance produced by moving objects before the objects move.

‘

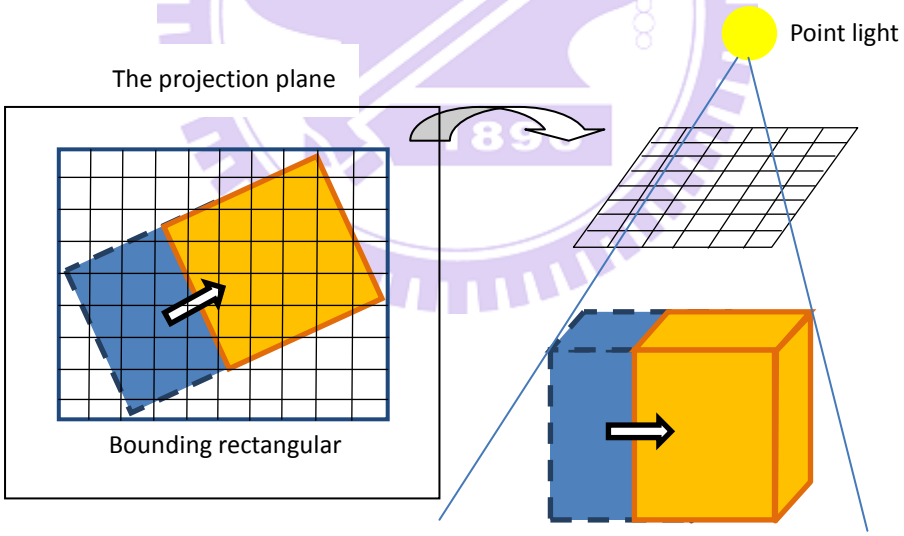


Figure 3.4 (a) Set the bounding rectangular from the point light's view. (b)Emit the compensation photons in the region of moving volume.

3.4 Update Hit Points

After we trace the compensation photons, we move the objects. In Section 3.2, we mentioned that the most different part between two frames is the indirect radiance bounced to the moving volume of the moving objects. The hit points in the moving volume are almost changed. If we use the compensation photons to eliminate the radiance in this volume, it would be inefficient. Due to this, we directly update the information of hit points in the moving volume, and the information of other hit points remains the same. There is a black area in Figure 3.5 that shows the hit points we update.

Note that, during updating the hit points, we still loop through all the hit points to tag which hit point we have updated. Because the radiance of these hit points is set to zero, we should compensate these hit points more radiance then others hit points in next step. We use a list to store the serial numbers of the hit point we have updated.

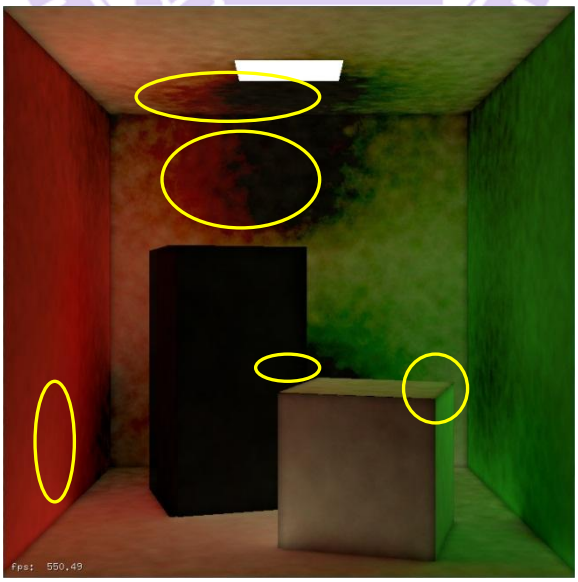


Figure 3.5 The image after emitting the compensation photon and update the hit points.

3.5 Compensate The Scene

After we move the objects, there are two parts in the scene where we should compensate in this pass. One is the indirect radiance bounced to the moving volume. The other is the indirect radiance produced by moving objects. To compensate the indirect radiance produced by the moving objects, we emit the photons with the path same as the path of the compensation photons as the green dashed region shown in Figure 3.6.

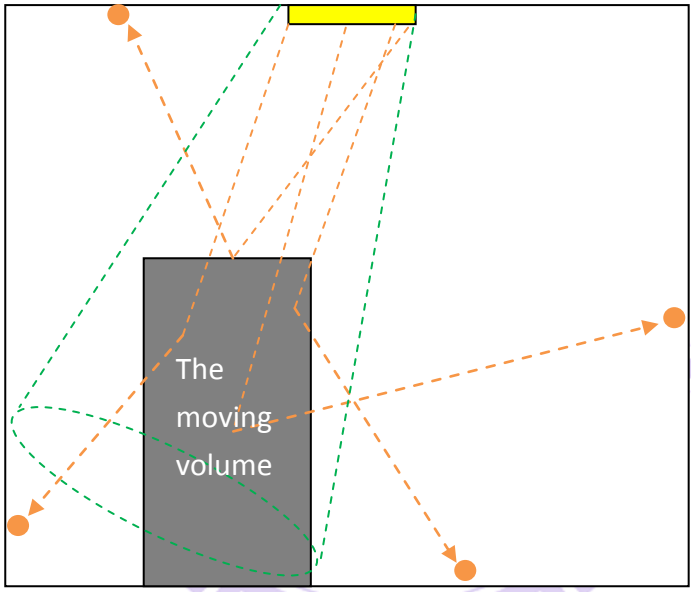


Figure 3.6 The tracing paths are used to trace the photons to compensate the indirect radiance.

To compensate the indirect radiance bounced to the moving volume, we can not just emit the positive radiance photons with the path same as the path of the compensation photons. In Figure 3.7, the orange dashed lines represent the photon tracing path inside the region, and the blue dashed lines represent the photon tracing path outside the region. It shows that there are a lot of photons accumulated by the hit point and the tracing path outside the region.

Before we move the objects, we emit the testing photons into the scene and store the

directions of the photons emitted from light source. In Section 3.3, we stored the serial numbers of the hit points we have updated. We only use these stored hit points to accumulate the testing photons in the gathering pass. Therefore, we can get the directions from these accumulated testing photons. Then we can emit the photons to compensate the radiance by these directions effectively. Besides, we only use stored hit points to accumulate the positive photons in gathering pass. In the progressive photon mapping, it repeats the photon tracing pass many times. If we use these directions repeatedly, it will cause that the photons are over accumulated by some hit points and there would be a lot of highlight in the image, as shown in Figure 3.10. To solve this artifact, we jitter the directions respectively within a cone with opening angle α of the cone in each photon tracing pass. According to the experiments, if α is too large, the indirect radiance is compensated inefficiently. And if α is too small, the artifacts still exist. So we define the region of α within $10°$. In Figures 3.8 and 3.9, we can observe that we compensate the indirect radiance of the moving volume by these directions more effectively.

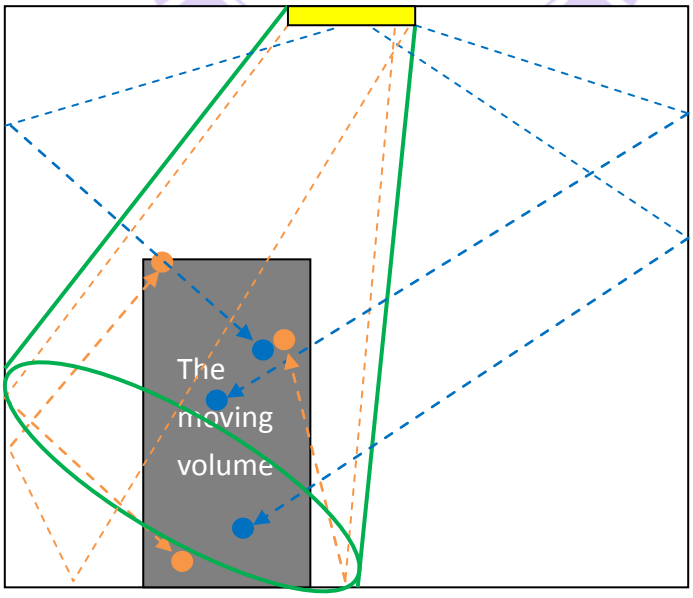


Figure 3.7 The tracing path of the photons, and the photons are accumulated by the hit points in the moving volume.

Figure 3.8 The scene is compensated by the photons which tracing path only within the bounding rectangular.

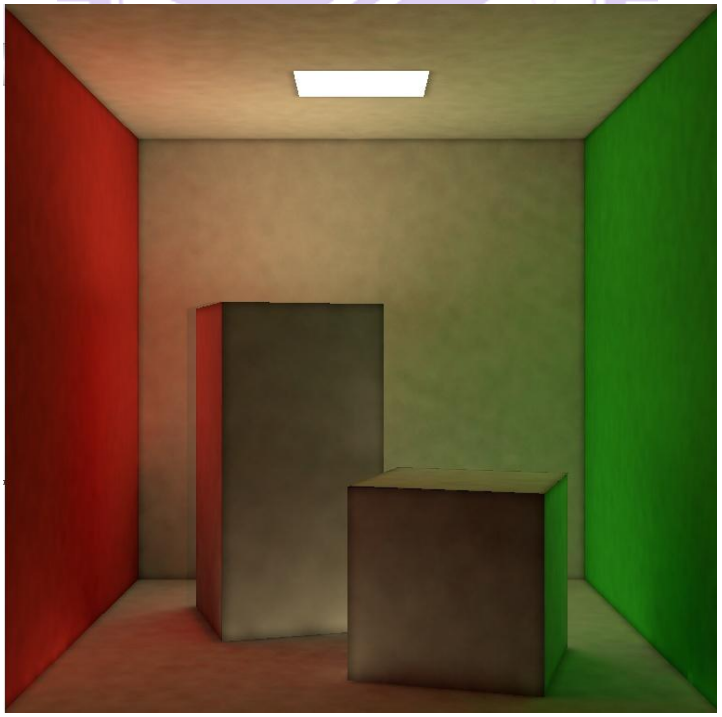


Figure 3.9 The scene is compensated by the photons which tracing path include the directions we record.

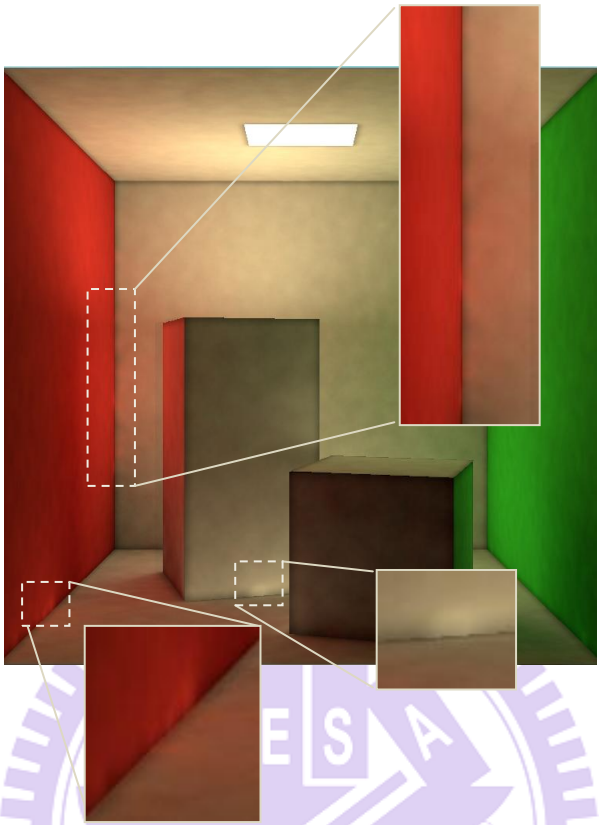


Figure 3.10 The artifact is caused by the directions of tracing photons without jittering.

3.6 Radiance Evaluation

In the progressive photon mapping, it emits positive radiance photons in each photon tracing pass. After each photon tracing pass, it accumulates the photons to evaluate the radiance by looping through all the hit points and render an image in gathering pass. In each gathering pass, it does three steps. The first step is radius reduction. Each hit point has a radius. While the number of photons accumulated within this radius increases, the radius needs to be reduced. The second step is flux correction. When a hit point receives new photons, it needs to accumulate the flux carried by those photons. So it needs to adjust this flux to take into account the radius reduction. The third step is radiance evaluation. By using the radius, the current flux multiplied by the BRDF, and the total number of emitted photons

in order to normalize the flux, it can evaluate the radiance at each hit point.

In our approach, after we emit the positive radiance photons in the photon tracing pass, the radiance evaluation in the following gathering pass is the same with the progressive photon mapping. But we should reformulate the radiance evaluation after we emit our negative power compensation photons in each photon tracing pass.

For radius reduction, we reformulate this to radius addition. When the hit points accumulate the compensation photons, we reduce the number of the photons accumulated within the radius. The number is computed as follows:

$$\widehat{\mathrm{N}}(\mathrm{x}) = \mathrm{N}(\mathrm{x}) - \alpha\mathrm{M}(\mathrm{x}) \tag{5}$$

where $\mathrm{N}(\mathrm{x})$ is the current number of the photons accumulated, $\mathrm{M}(\mathrm{x})$ is the additional number of the photons accumulated, and $\alpha, 0 < \alpha < 1$, is only a fraction.

By reducing the number of the accumulated photons, we can increase the radius as follows:

$$\widehat{\mathrm{R}}(\mathrm{x}) = \mathrm{R}(\mathrm{x}) + d\mathrm{R}(\mathrm{x}) = \mathrm{R}(\mathrm{x})\sqrt{\frac{\mathrm{N}(\mathrm{x})-\alpha\mathrm{M}(\mathrm{x})}{\mathrm{N}(\mathrm{x})-\mathrm{M}(\mathrm{x})}} \tag{6}$$

If we over accumulate the negative power photons, it may cause the $\mathrm{R}(\mathrm{x})$ to turn to negative. We should clamp the value of the radius to one, if the value of the radius is less than zero.

For flux correction, we reformulate the flux evaluation as follows:

$$\tau_{\widehat{\mathrm{N}}}(x, \vec{\omega}) = \tau_{\mathrm{N}-\mathrm{M}}(x, \vec{\omega})\frac{\mathrm{N}(\mathrm{x})-\alpha\mathrm{M}(\mathrm{x})}{\mathrm{N}(\mathrm{x})-\mathrm{M}(\mathrm{x})} \tag{7}$$

where the quantity τ is represented as flux. If we over accumulate the negative radiance photons, it may cause the value of flux to be less than zero. We clamp the value of flux to zero, if the value of flux is less than zero.

For radiance evaluation, we use the Equation 4. mentioned in Section 2.2.

Chapter 4
Implementation and Results

Our results are rendered on a desktop PC with Intel Core i7 930 CPU 2.8GHz and NVIDIA GeForce GTX 480 video card. All results are rendered at 768 x 768 pixels.

In the progressive photon mapping, direct illumination and indirect illumination are rendered by tracing the photons in photon tracing pass. In our approach, we render direct illumination by using a similar phong lighting model based on the demo of progressive photon mapping of NVIDIA Optix. We only implement our algorithm on indirect illumination.

GPUs are best at exploiting very high degrees of parallelism, and ray tracing fits that requirement perfectly. Optix is a simple but powerful abstract model of a ray tracer running entirely on the NVIDIA CUDA compute architecture. So we implement our approach by OpenGL, and the major computation part is performed by Optix engine including the ray tracing pass, the photon tracing pass, and the gathering pass. To store the data of each hit point and each photons, we use the buffer stored in GPU memory designed by Optix. This buffer can be accessed by CPU and GPU, and then we can access the data efficiently. In program, when we render the scene, we only compute the photon map on CPU. Other works are performed by GPU.

When sending mesh data into Optix, we should construct the geometry group in Optix context first. As shown in Figure 4.1, a geometry group can contain many geometry instances.

Each geometry group needs to have an acceleration object assigned for ray traversal to intersect. Each geometry instance represents a coupling of a single geometry node with a set of materials. Each geometry node is bounded with intersect program and bounding program. In intersect program, we compute the position of intersection of ray and geometry. In bounding program, we bound primitive data. In any-hit program and closet-hit program, we can trace the ray to obtain the external material such like shadow or indirect light. The difference between any-hit and closet-hit is using the ray payload or not. We use closet-hit program to obtain the information of ray payload such like color, normal. And any-hit program can only return hit geometry or not.

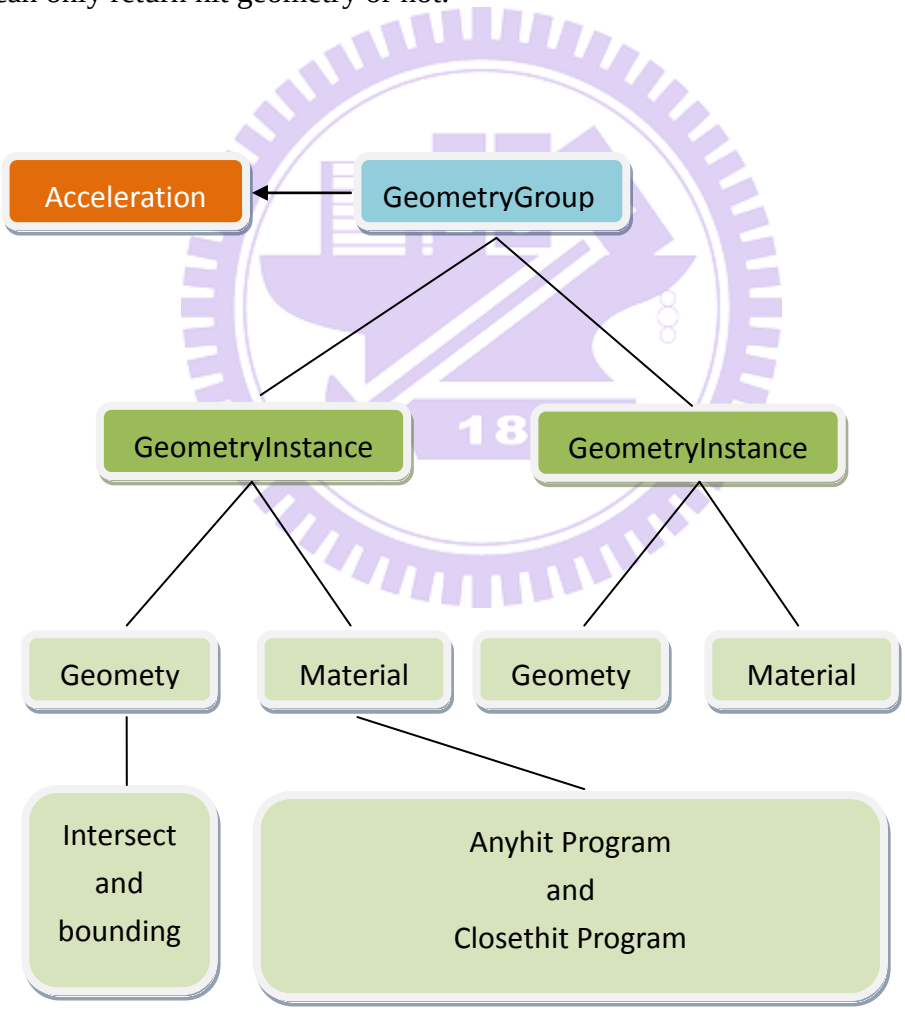


Figure 4.1 Optix data structure

We render the scenes by the progressive photon mapping all in 15 seconds before the objects move. After the objects move, we spend 8 seconds rendering the next frame. In order to highlight our contribution, we show comparison images with the same performance. One uses our algorithm to render the scene. The other uses the progressive photon mapping to render the scene with same performance.

Figure 4.3, 4.6, and 4.9 display the moving objects moved in the scenes rendered using the progressive photon mapping. Figure 4.4, 4.7, and 4.10 display the moving objects moved in the scenes rendered by our algorithm. Figure 4.2, 4.5, and 4.8 display the ground truth of the scenes. Figure 4.2 - 4.4 display basic cornell box scene. Figure 4.5 - 4.7 display cornell box scene containing a complex model. Figure 4.8 - 4.10 display the effect of caustic in the wedding -band scene.

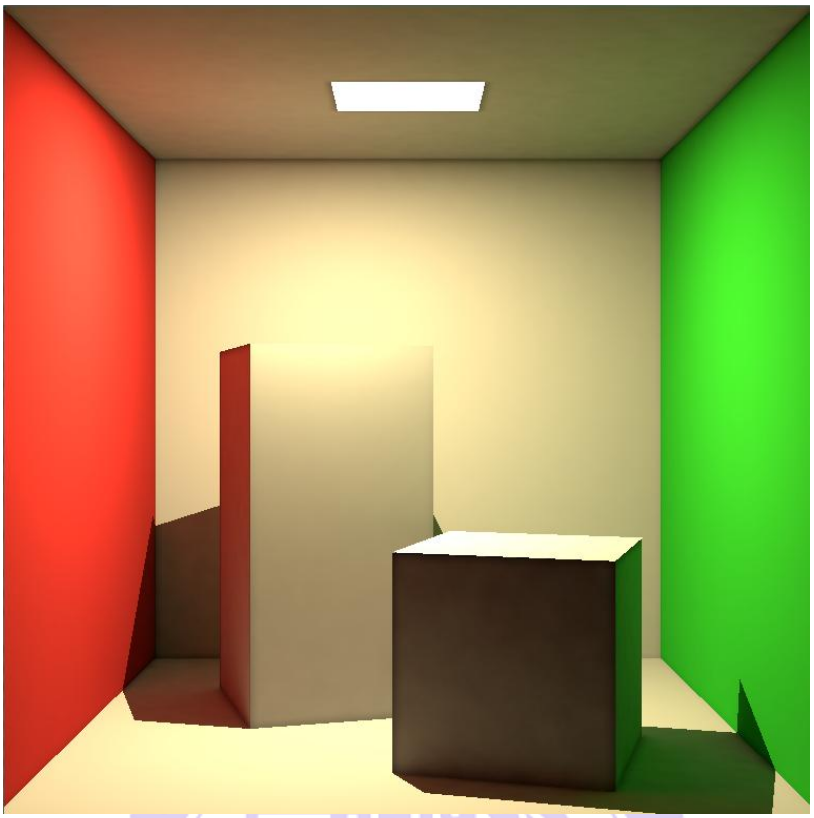


Figure 4.2 The ground truth of tall box moving in cornell box scene.

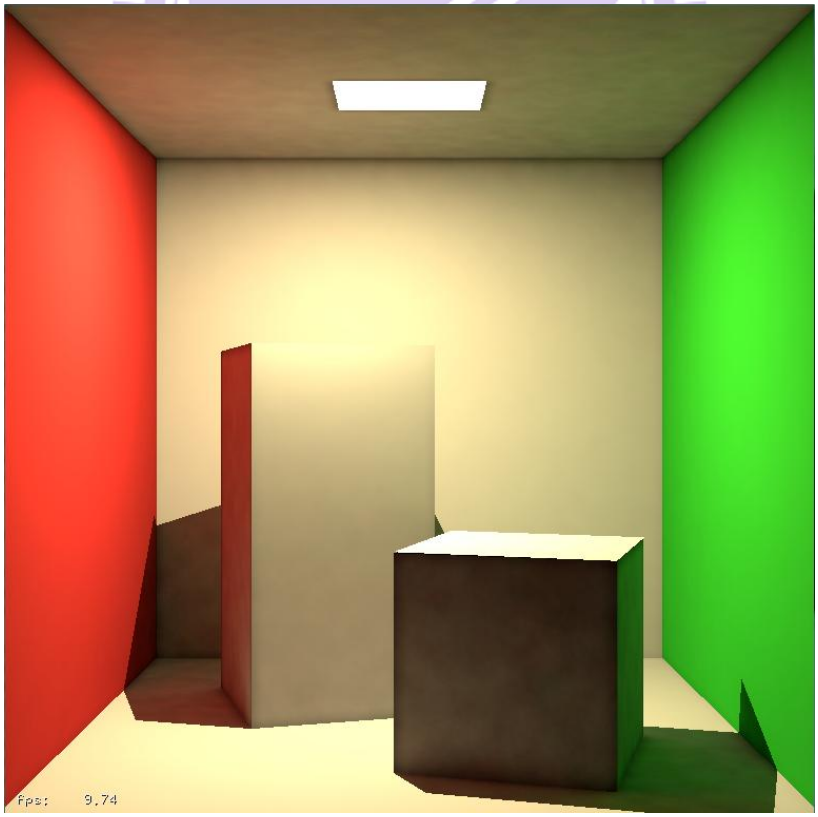


Figure 4.3 Tall box moves in cornell box scene using PPM.

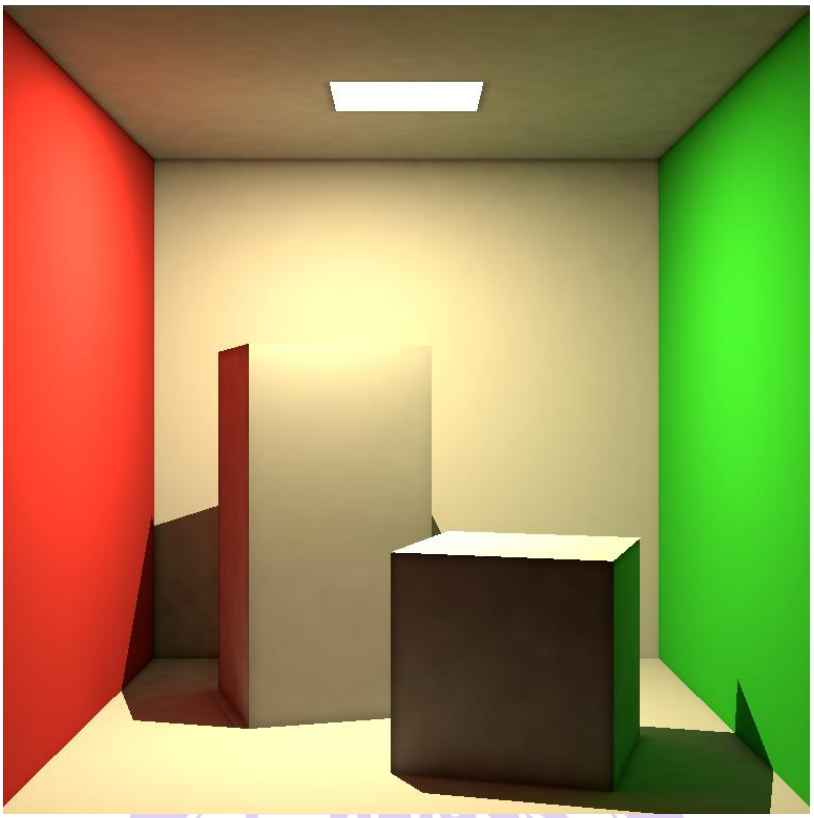


Figure 4.4 Tall box moves in cornell box scene using our method.

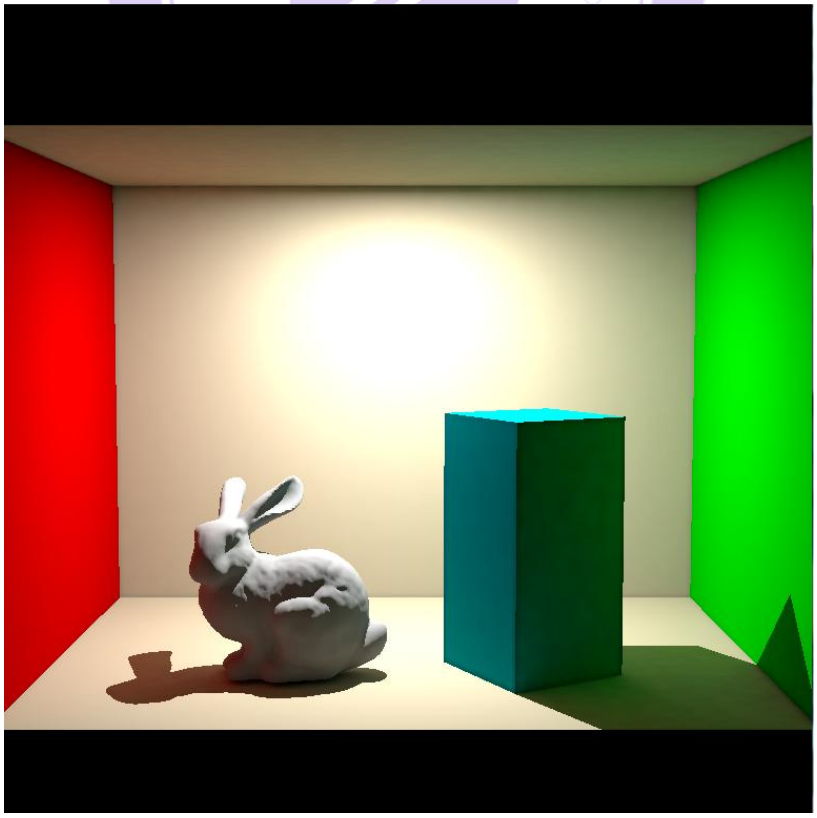


Figure 4.5 The ground truth of bunny moving in cornell box scene.

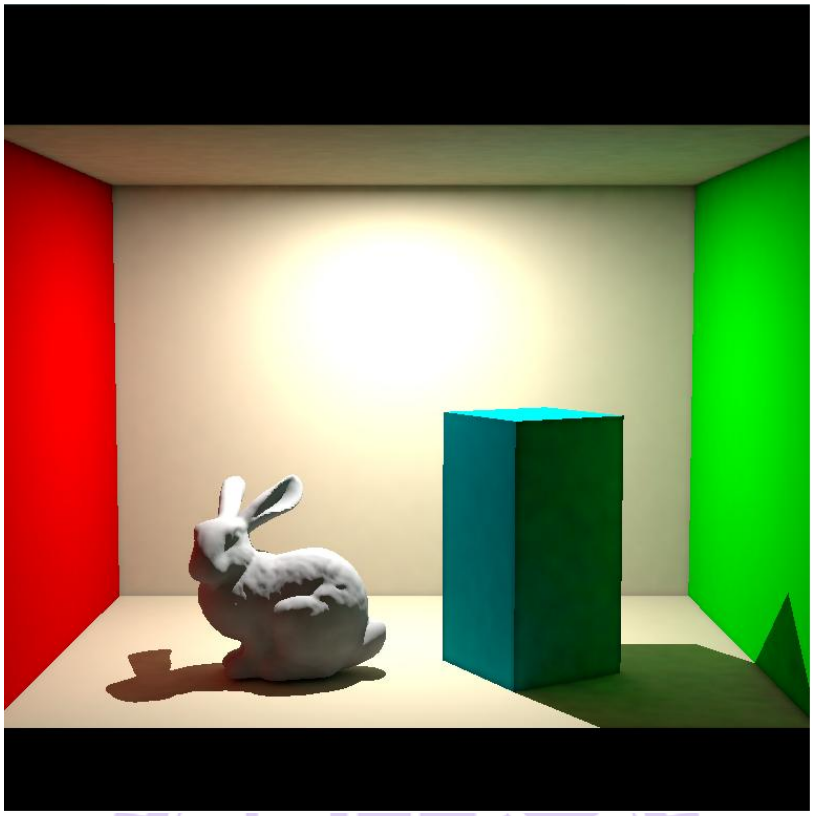


Figure 4.6 Bunny moves in cornell box scene using PPM.

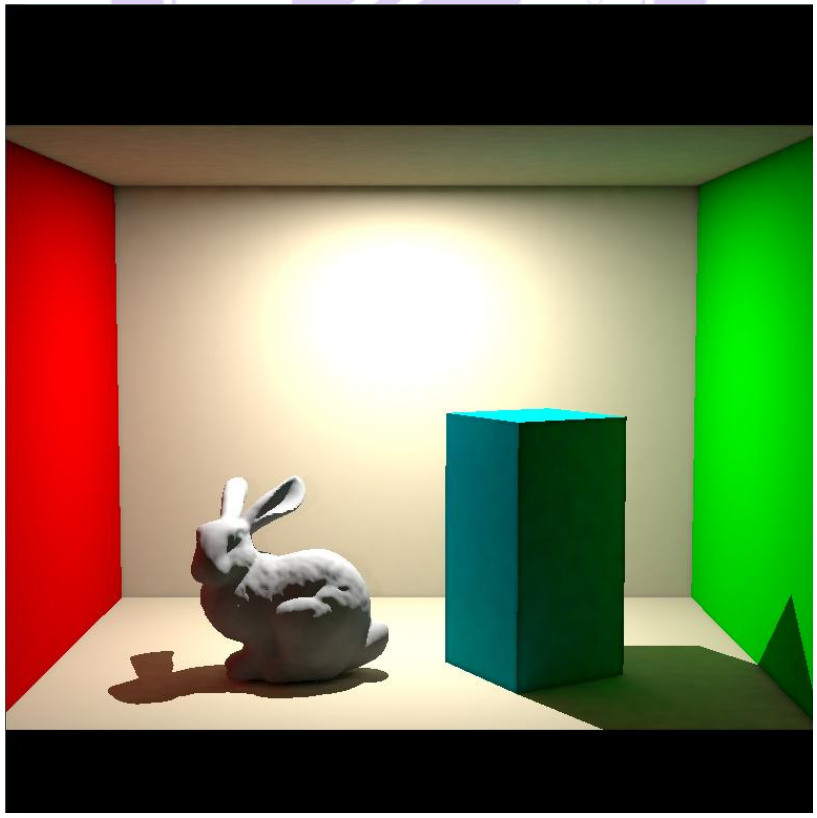


Figure 4.7 Bunny moves in cornell box scene using our method.

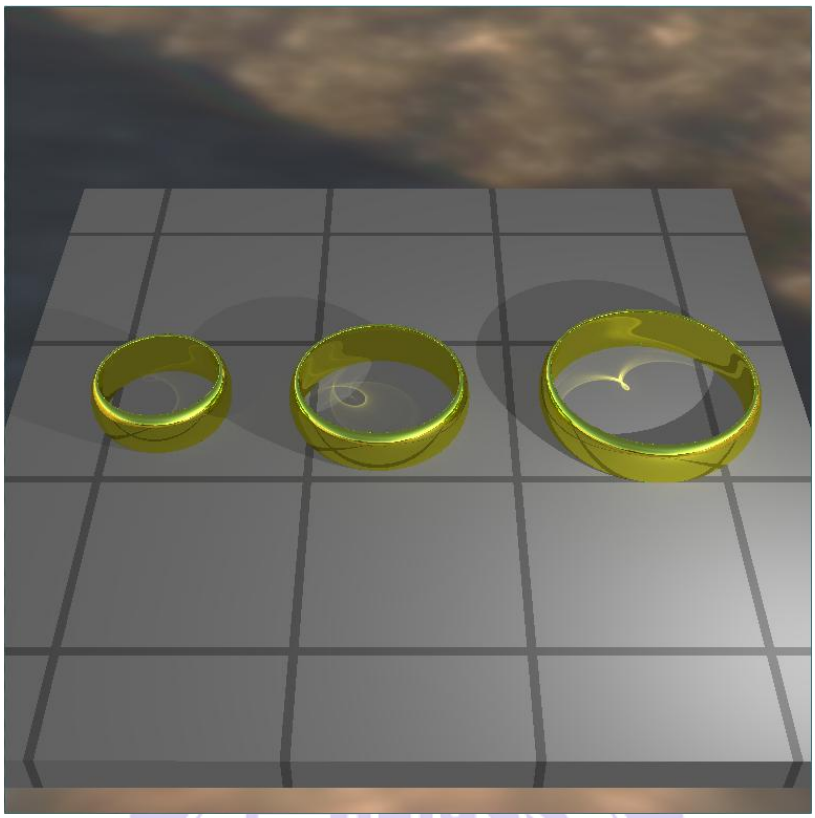


Figure 4.8 The ground truth of middle ring moving in wedding-band scene.

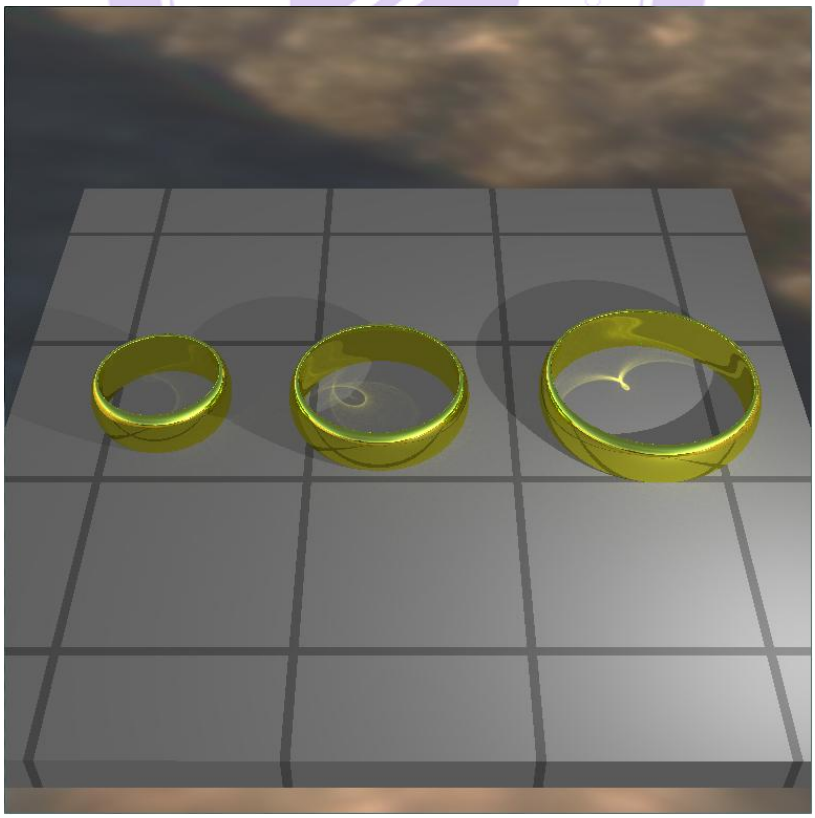


Figure 4.9 Middle ring moves in wedding-band scene using PPM.

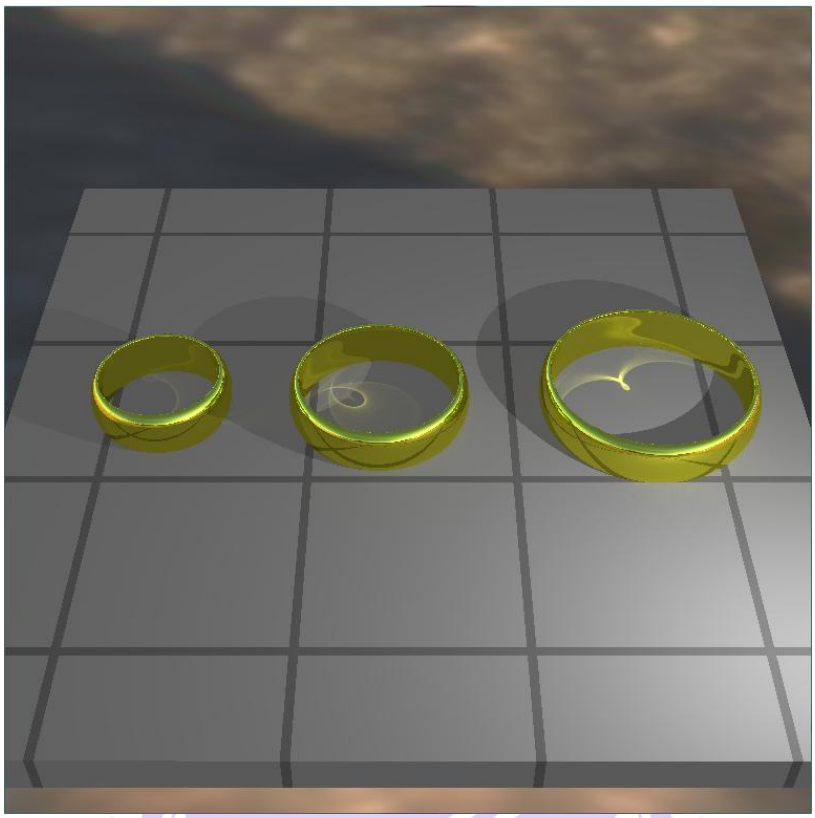
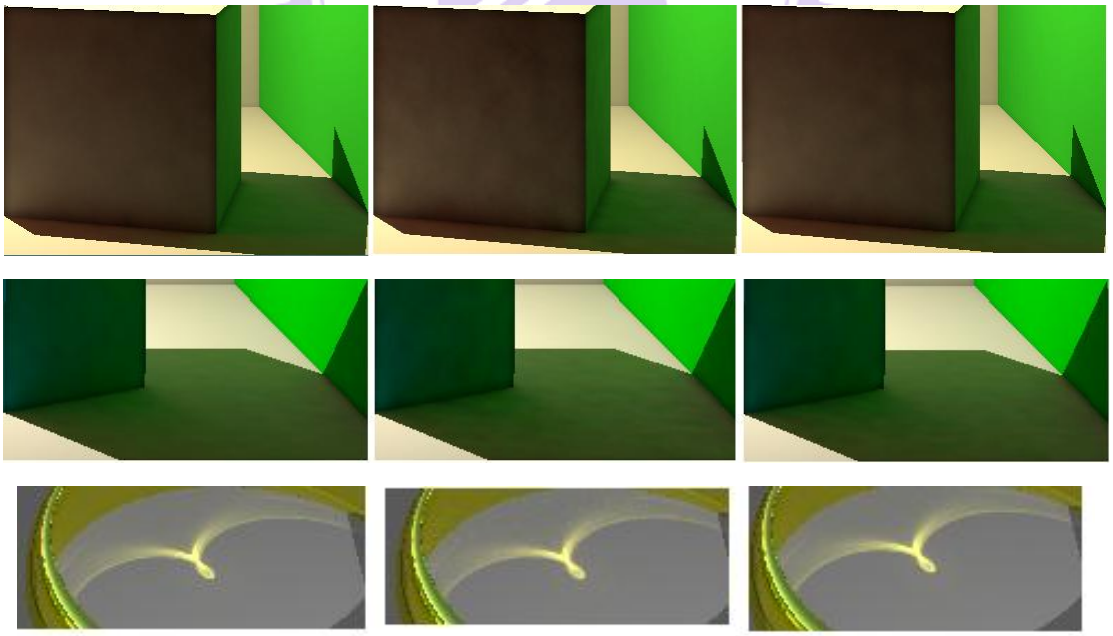


Figure 4.10 Middle ring moves in wedding-band scene using our method.



Ground truth PPM Our result

Figure 4.11The comparison of figures 5.1 to 5.9

In figure 4.11, we can observe that our result can render the indirect illumination part smoother. We can keep the radiance which the moving object did not affect, so we do not need to emit the same number of photons to render. Our result demonstrates that we can get more indirect illumination with same performance. Note that, the major concept of our algorithm is to keep the illumination between two frames. In others word, the moving objects make smaller influence to scene, we can keep more illumination between two frames.

Table 1 and 2 shows the experimental result of our rendering algorithm. We can observe that the effect of caustic in wedding-band scene needs more photons to render than other scenes.

Scene	Triangles	Compensation photons (radius)	Positive power photons (radius)	Valid photons	Time
Cornell box	15	1310k(400)	7864k(400)	5854k	8s
bunny	69473	918k(2)	8126k(2)	5373k	8s
Wedding-band	41196	3146k(0.25)	3277k(0.25)	893k	4s

Table 4.1：Detail statistics for scenes with our algorithm. The "radius" is the initial radius of hit point.

Scene	Triangles	Total Photons (radius)	Valid photons	Time
Cornell box	15	10485k(400)	6702k	8s
bunny	69473	9961K(2)	6777k	8s
Wedding-band	41196	8847k(0.25)	247k	4s

Table 4.2：Detail statistics for scenes with PPM.

Ground truth PPM Our result

Figure 4.12 The comparison of indirect illumination part

In figure 4.12, we can obviously observe the difference between our result and the result of the progressive photon mapping. The comparison part has no object moved. So this part keeps the same quality with the first frame. In the progressive photon mapping, accumulating more photons, the image will be smoother. The difference between the ground truth and our algorithm is smaller than the difference between the ground truth and the progressive photon mapping.

Chapter 5
Conclusion and Future Works

In this thesis, we propose a GPU-based global illumination algorithm that speed up the progressive photon mapping in dynamic scenes. We propose a new idea to keep the same illumination between frames. For progressive photon mapping, we add a new type of photon called the compensation photon. Because we add a new type photon, it needs to reformulate the radiance evaluation to evaluate the radiance of compensation photon.

In order to trace positive power photons to compensate the scene, we find out two tracing mechanism on tracing path. One is tracing the photons towards the moving volume of the moving objects. The other is to use the directions we record to emit the photons to compensate the indirect radiance bounced by the moving volume. Finally we use these two tracing mechanism to compensate the scene.

In the future we would improve the compensating part. If we can compensate the indirect radiance of the moving volume effectively, we can spend less time rendering the scene. In addition to progressive photon mapping, we would like to speed up other global illumination in dynamic scenes such like instant radiosity, ray tracing. Our algorithm is performed based on fixed perspective. We would extend our algorithm to dynamic perspective. For application, we would apply our algorithm on editing program.

Reference

[1] APPEL, A. 1968. Some techniques for shading machine renderings of solids. In AFIPS ’68 (Spring): Proceedings of the April30–May 2, 1968, spring joint computer conference, ACM, New York, NY, USA, 37–45.

[2] COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In Proceedings of SIGGRAPH, 137–145.

[3] DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In Proc. of ACM SI3D ’05, ACM, New York, NY, USA, 203–231.

[4] DACHSBACHER, C., STAMMINGER, C., DRETTAKIS, G., andDURAND, F. 2007. Implicit Visibility and Antiradiance for Interactive Global Illumination. ACM Transactions on Graphics(Proc. of SIGGRAPH) 26, 3.

[5] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. ACM Transactions on Graphics 28, 5 (2009), 1. 1, 2.

[6] HACHISUKA, T., OGAKI, S., AND JENSEN, H. W. 2008. Progressive photon mapping. ACM Transactions on Graphics (SIGGRAPH Asia Proceedings) 27, 5, Article 130.

[7] HERZOG, R., HAVRAN, V., KINUWAKI, S., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2007. Global illumination using photon ray splatting. In Eurographics 2007, vol. 26, 503–513.

[8] JENSEN, H. W. 1996. Global illumination using photon maps. In Proceedings of the eurographics workshop on Rendering techniques’96, Springer-Verlag, London, UK, 21–30.

[9] KELLER, A. 1997. Instant radiosity. In SIGGRAPH ’97: Proc. of the 24th annual conference on comp. graph. and interactive techniques, ACM Press/Addison-Wesley

Publishing Co., New York, NY, USA, 49–56.

[10] LANDIS H. 2002. Production-ready global illumination. Course notes for SIGGRAPH 2002 Course 16, RenderMan in Production.

[11] MA, V. C. H., AND MCCOOL, M. D. 2002. Low latency photon mapping using block hashing. In HWWS ’02: Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graph. hardware, Eurographics Association, Aire-la-Ville, Switzerland, 89–99.

[12] MCGUIRE, M., AND LUEBKE, D. 2009. Hardware-Accelerated Global Illumination by Image Space Photon Mapping. In Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics.

[13] PARKER, S. G., BIGLER, J., DIETRICH, A., FRIEDRICH, H., HOBEROC, K J., LUEBKE, D., MCALLISTER, D., MCGUIRE M., MORLEY, K., ROBISON, A., STICH, M. 2010. Optix: A general purpose ray tracing engine. ACM Transactions on Graphics.

[14] PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, Eurographics Association, 41–50.

[15] RITSCHEL, T., GROSCH, T., KIM, M., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. ACM Trans. Graph. 27, 5, 1–8.

[16] ROBISON, A., AND SHIRLEY, P. 2009. Image space gathering. In Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conferenceon High Performance Graphics.

[17] SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3-d shapes. SIGGRAPH Comput. Graph. 24, 4, 197–206.

[18] WACHOWICZ, P. 2011. Accelerating Photon Mapping with Photon Flipping and Invalidity Photons. Master thesis, University of Amsterdam.

[19] WANG, R., WANG, R., ZHOU, K., PAN, M., AND BAO, H. 2009. An efficient gpu-based approach for interactive global illumination. ACM Trans. Graph. 28, 3, 1–8.

[20] WEISS, M., AND GROSCH, T. Stochastic Progressive Photon Mapping for Dynamic Scenes. EUROGRAPHICS 2012 / P. Cignoni, T. Ertl. (Guest Editors). Volume 31 (2012), Number 2.