BLUE NOISE VIRTUAL POINT LIGHTS

FOR GLOBAL ILLUMINATION

by

Mohammad Montazerolzohour

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

August 2017

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Mohammad Montazerolzohour**

has been approved by the following supervisory committee members:

Cem Yuksel	, Chair	**05/04/2017** Date Approved
Charles Hansen	, Member	**05/04/2017** Date Approved
Erik Brunvand	, Member	**05/04/2017** Date Approved
Ladislav Kavan	, Member	**05/04/2017** Date Approved

and by **Ross Whitaker**, Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Virtual point lights (VPLs) provide an effective solution to global illumination computation by converting the indirect illumination into direct illumination from many virtual light sources. This approach results in a less noisy image compare to Monte Carlo methods. In addition, the number of VPLs to generate can be specified in advance; therefore, it can be adjusted depending on the scene, desired quality, time budget, and the available computational power.

In this thesis, we investigate a new technique that carefully places VPLs for providing improved rendering quality for computing global illumination using VPLs. Our method consists of three different passes. In the first pass, we randomly generate a large number of VPLs in the scene starting from the camera to place them in positions that can contribute to the final rendered image. Then, we remove a considerable number of these VPLs using a Poisson disk sample elimination method to get a subset of VPLs that are uniformly distributed over the part of the scene that is indirectly visible to the camera. The second pass is to estimate the radiant intensity of these VPLs by performing light tracing starting from the original light sources in the scene and scatter the radiance of light rays at a hit-point to the VPLs close to that point. The final pass is rendering the scene, which consists of shading all points in the scene visible to the camera using the original light sources and VPLs.

To my wife, Arezou,
to my mother, Shahnaz,
to my father, Reza,
to my sister, Nahid,
and to my brothers, Alireza, Hamidreza, Farid, and Morteza,
without whom none of my success would be possible.

TABLE OF CONTENTS

LIST OF FIGURES

Figures

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Cem Yuksel, for his continuous support, knowledge, wisdom, and kindness throughout my education at The University of Utah.

My sincere thanks also go to Ian Mallet for his valuable guidance during this research.

CHAPTER 1

INTRODUCTION

Image synthesis, which is one of the core functions of Computer Graphics, is creating images from some description data. Photo-realistic image synthesis aims to simulate the interaction of light with objects in the real world. The process of rendering an image from a scene file describing all objects, lights, materials, etc. is called rendering. The goal of this process is finding the colors of points in the scene which are visible to the camera. To find the color of each point, we need to compute the outgoing illumination from that point to the camera.

1.1 Global Illumination with Virtual Point Lights (VPLs)

The incoming illumination at a point x can be separated into direct illumination (coming directly from the light sources), and indirect illumination that indicates the reflected light from other points in the scene. Global illumination (GI) refers to the combined illumination including both direct and indirect illumination (Figure 1.1) and computing global illumination can be difficult and time consuming, but it is usually the indirect illumination component that is computationally very expensive.

Computing global illumination is a hard problem because light can reflect from almost any object in a scene and that object will act as a light source (by reflecting the

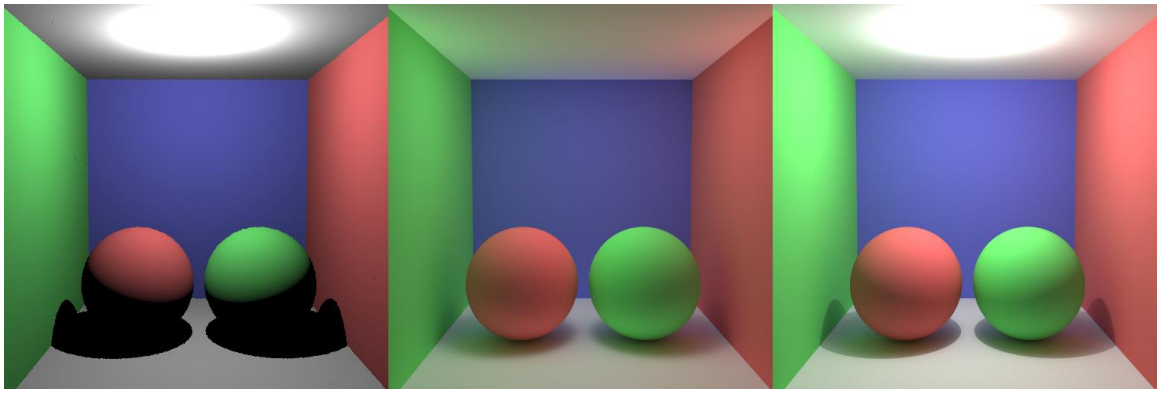


Figure 1.1: Direct illumination, indirect illumination, global illumination

incoming light shining on it) in the scene, even if it is not a light emitter itself. Therefore, to compute the correct illumination at a surface point visible by a camera, we need to compute all light coming from any direction from both light sources in the scene and other objects reflecting light to that point, which, at least in theory, requires a recursive computation. At each light bounce from an object surface, light can be reflected in any direction, depending on the material properties of the surface, and the reflected light can continue to bounce off of other surfaces in the scene in arbitrary directions. Computing all possible light paths in a scene is very difficult and needs a lot of time and computational power.

One approach for computing global illumination is converting indirect illumination into direct illumination from multiple virtual point lights (VPLs), which was first introduced by Keller [1997]. The idea is that instead of computing indirect light coming from all directions in the scene, we can render a scene in two passes. First, we do light tracing, starting from light sources in a scene and place a VPL, storing the radiant intensity of the light ray, wherever the light ray hits a surface. In the second pass, the objects are rendered using only direct illumination from these VPLs and light sources. In

fact, by applying this technique, we approximate the global illumination with only direct illumination. As a result, we can estimate the global illumination and generate an output image with less noise compare to other techniques.

In this thesis, I investigate different methods of placing VPLs for improving the convergence of the lighting computation.

Thesis Statement: Uniformly distributing VPLs to the surfaces that indirectly illuminate the points visible to the camera can produce improved rendering quality for computing global illumination using VPLs.

The particular algorithm we investigate consists of three different passes:

In the first pass, we randomly generate lots of VPLs in the scene and place them in the positions that can contribute to the final rendered image by tracing light paths starting from the camera. Then we remove a considerable portion of these VPLs using a sample elimination method [Yuksel 2015] to get a subset of VPLs that are uniformly distributed all over part of the scene that is relevant for the camera view. The goal is placing VLPs spatially evenly over the part of the scene that is indirectly visible to the camera.

The second pass is to estimate the radiant intensity of these VPLs by doing light tracing from original light sources in the scene and scattering the radiance carried by each light ray to the VPLs near the point it hits.

The final pass is rendering the scene that consists of shading all points in the scene which are visible to camera using original light sources in the scene and the VPLs we generated in the first pass. The illumination from the original light sources corresponds to direct illumination, and the illumination from the VPLs corresponds to indirect illumination, as in all VPL-based global illumination computation methods.

CHAPTER 2

BACKGROUND

Rendering Equation (Equation (2.1)) is an integral over all possible incoming light directions that can be reflected towards a specific direction, which is, directly or indirectly, visible to the camera. The outgoing light at a point includes all emitted and reflected light by the surface point, which depends on surface geometry, material, and lights in the scene.

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i)(\omega_i \cdot n) d\omega_i, \qquad (2.1)$$

where $L_o(x, \omega_o)$ is the total radiance of light going out in direction ω_o, from a point x, $L_e(x, \omega_o)$ is emitted radiance, Ω is the unit hemisphere aligned with the surface normal vector n containg all possible values for ω_i, $f_r(x, \omega_i, \omega_o)$ is bidirectional reflectance distribution function (BRDF) with respect to light coming from ω_i and going to ω_o at point x, $L_i(x, \omega_i)$ is incoming radiance, ω_i is negative incoming light direction, and n is the surface normal at point x.

Rendering Equation was introduced by Kajiya [1986] and the goal of all physically-based rendering algorithms is to compute this equation to estimate the outgoing radiance at any point. Solving this equation is the aim of photo-realistic rendering and there are several different methods proposed by graphics researchers over the years, including

finite element methods like the radiosity algorithm [Cohen et al. 1993], and Monte Carlo methods like path tracing [Kajiya 1986], photon mapping [Jensen & Christensen 1995], and Metropolis light transport (MLT) [Veach & Guibas 1997], just to give a few examples.

2.1 Rendering with Virtual Lights

The VPL-based global illumination computation methods place many secondary light sources in a scene and then approximate the light transport using only direct illumination from these secondary light sources. This approach results a less noisy image compare to Monte Carlo methods. In addition, the number of VPLs to generate can be specified in advance, and therefore, it can be tweaked depending on the scene, desired quality, time budget, computational power budget, etc.

To approximate the illumination at a point, we can use Equation (2.2) which is derived from the rendering equation.

$$L_o(x, \omega_o) \approx L_e(x, \omega_o) + \sum_{i}^{N} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) cos\theta \qquad (2.2)$$

where θ is the angle between surface normal at point x and the incoming light direction, N is the number of VPLs, and L_i is incoming illumination from VPL i. L_i can be computed as

$$L_i = I_i \, cos\phi \, \frac{1}{d^2}, \qquad (2.3)$$

where I_i is the intensity of VPL i, ϕ is the angle between surface normal where the VPL is placed and the light direction at point x, and d is the distance between point x and

VPL's position. This equation makes the VPL approximate a Lambertian emitter at the surface point it is placed, thereby emulating diffuse reflection of indirect illumination off of the surface.

Prior work on rendering with virtual lights can be grouped into two categories. The first category contains the techniques that are developed for approximating global illumination by computing direct illumination from many virtual lights. The second category consists of techniques that aim to efficiently render a scene that contains a large number of (virtual) light sources.

2.2 Approximating Global Illumination Using Virtual Lights

Instant Radiosity [Keller 1997] is a rendering algorithm introduced by Keller and it is the core of VPL-based global illumination methods. Instant Radiosity combines the advantages of quasi-random walk and Rendering Equation to generate an output image by rendering a scene to several buffers, assuming only diffuse surfaces, and then generating the final rendered image by summing up the buffers into an accumulation buffer.

To render a scene, initially N particles are selected to start off the light source randomly or using a quasi-Monte Carlo sequence. For each particle, the scene is rendered into a buffer, assuming the particle is the only point light source in the scene. Then, kN rays are shot into the scene and at each hit-point, the particle is attenuated by the BRDF of the surface at the hit-point and the scene is rendered again for each new particle. In the next step, k^2N particles are generated and sent into the scene; this process is repeated until a maximum path length is reached. At the end, all rendered buffers are summed up

and then divided by number of buffers, which is equal to number of particles created.

One difficulty with the VPL-based global illumination methods is that they generate a large number of light sources and illuminating a scene using a large number of light sources can be computationally expensive. This problem is often referred to as the many-lights problem.

Since the introduction of Instant Radiosity, many researchers proposed new techniques based on many-lights methods because of their artifact-free outputs and scalability [Dachsbacher et al. 2014]. The many-lights techniques can be summed up in two main categories: the techniques to solve the global illumination problem using many virtual lights, and the techniques to render a scene which has many-lights faster and more efficiently.

Wald et al. [2002] introduced a new parallel global illumination algorithm with highly efficient sampling and scalability. They achieve interactive rates by precomputing point lights, using photon map only for caustics, interleaved sampling, and using an efficient randomized quasi-Monte Carlo integration.

Wald et al. [2003] achieved interactivity for highly occluded scenes by introducing an importance sampling technique that is efficient in those types of scenes. In this method, an estimate of each light's importance, computed by an eye path tracer, is used to steer the rendering budget more toward the most important light sources in the scene.

Dachsbacher & Stamminger [2005] approximate global illumination by calculating the incoming illumination from some VPLs generated based on the shadow map, without considering the occlusion. For each shading pixel, the corresponding pixel in the shadow map is found. Then, hundreds of pixels around the shading pixel in the shadow map are

selected and these pixels are treated as VPLs. The direct illumination coming from these VPLs roughly approximate the indirect illumination for the shading pixel.

Segovia et al. [2006] generate VPLs both from the camera and from the light sources and using different estimators, a subset of VPLs with a density proportional to the power that is received by the camera is chosen. Georgiev & Slusallek [2010] pick a subset of VPLs by a sample rejection method and Segovia et al. [2007] sample VPLs by a modified MLT approach.

Hašan et al. [2009] introduced virtual spherical lights (VSLs) to capture the illuminations that cannot be captured by VPL easily. The advantage of using sphere lights, instead of point lights, is that they can be used to prevent loss of energy, due to using diffuse VPLs, in the scenes with glossy surfaces.

VPLs are not suitable for representing glossy materials, because VPLs usually take only one direction to account and this is a very small fraction of light bouncing back from a glossy surface. Therefore, the VPLs need to represent multiple light paths to be efficient for glossy surfaces. In contrast to regular VPLs, Rich-VPLs [Simon et al. 2015] represent a lot of light paths, so they can contribute more to simulate glossy surfaces (Figure 2.1).

To sample a light path X, vertex x_1 at camera and x_k at light source, we compute light transport by

$$T(X) = W(x_1)\left(\prod_{i=1}^{k-1} G_{x_i \leftrightarrow x_{i+1}}\right)\left(\prod_{i=2}^{k-1} f(x_i)\right)L(x_k), \qquad (2.4)$$

where $W(x_1)$ is the importance of the sensor, $G_{x_i \leftrightarrow x_{i+1}}$ is the geometry term when going from vertex x_i to vertex x_{i+1}, $f(x_i)$ is BRDF at surface point x_i, and $L(x_k)$ is the emitted radiance.

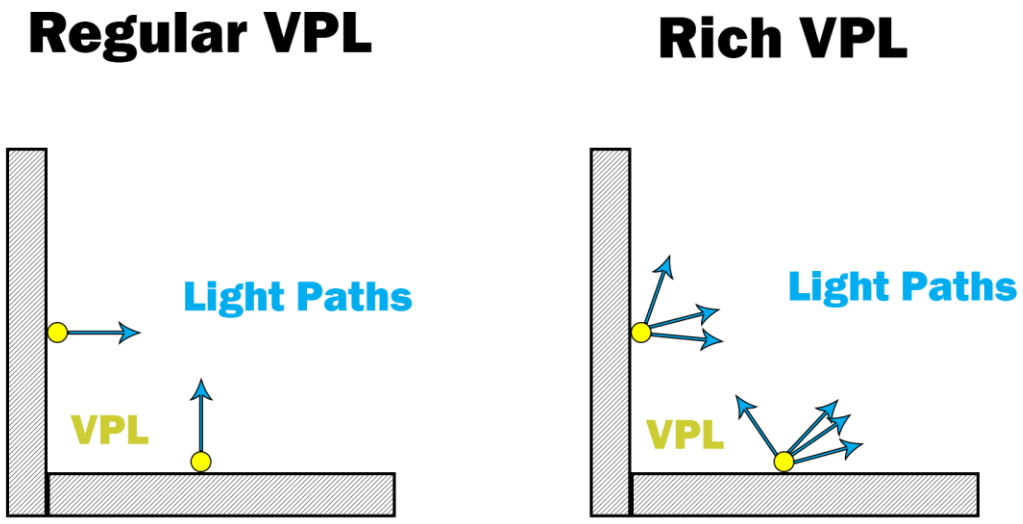


Figure 2.1: Regular VPLs vs. Rich-VPLs

To give an example, we consider a light path length of 3 or greater. The light transport can be written as multiplication of sensor importance $W(x_3)$ reaching x_3, BRDF $f(x_3)$, and the radiance $L(x_3)$ reaching x_3

$$T(X) = W(x_3) \cdot f(x_3) \cdot L(x_3), and$$

$$W(x_3) = W(x_1) \cdot G_{x_1 \leftrightarrow x_2} \cdot f(x_2) \cdot G_{x_2 \leftrightarrow 3}, and$$

$$L(x_3) = L(x_k)\left(\prod_{i=3}^{k-1} G_{x_i \leftrightarrow x_{i+1}}\right)\left(\prod_{i=4}^{k-1} f(x_i)\right) \quad (2.5)$$

This is the same as creating a VPL at x_3 with emission $f(x_3)L(x_3)$, illuminating surface point x_2 which is visible to camera (Figure 2.2). Therefore, Simon et al. proposed to sample VPL locations proportional to the product of the total importance reaching a

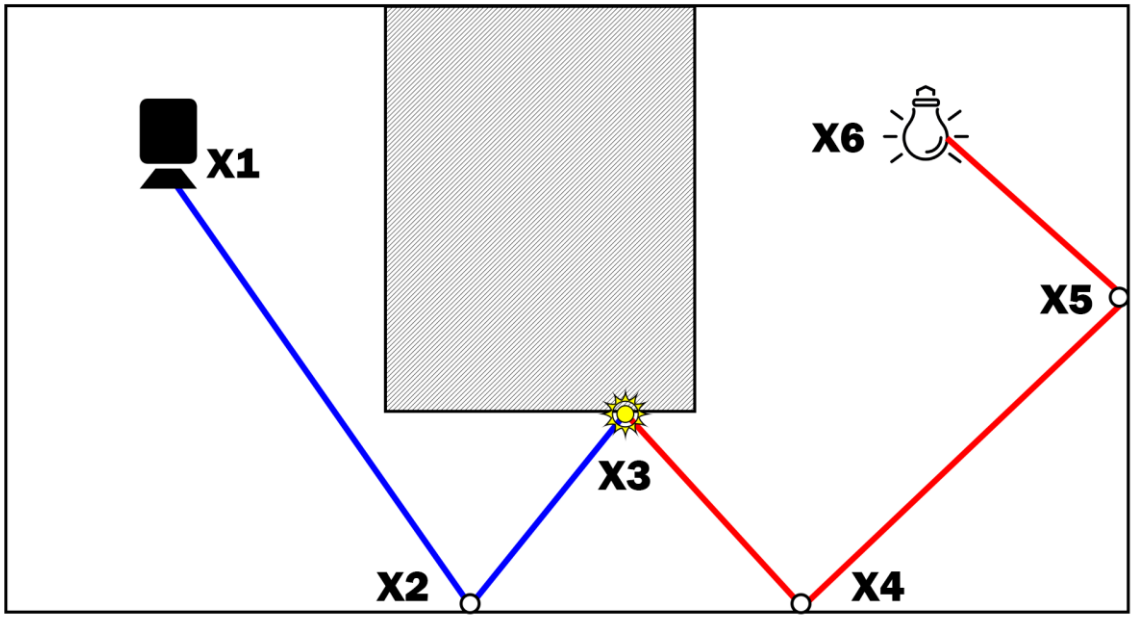


Figure 2.2: Transport path connecting the camera to the light source

surface point and the total incident radiance reaching that point.

To estimate the emission profile of each VPL, a photon map is created and then close photons are lumped together to estimate the light going out at a VPL. Because the light going out can have different directions and these directions can vary a lot, each Rich-VPL should store an approximation of all these light paths. This approximation is done differently for highly-glossy surfaces and moderately-glossy surfaces because the reflection of light is focused to a very small angle for highly-glossy surfaces, but that's not the case for moderately-glossy surfaces. In addition, VPLs on diffuse surfaces need only to store the outgoing light, because diffuse surfaces reflect the light almost evenly in any direction.

To place the VPLs in a scene, Rich-VPLs method takes a different approach: the position of the VPLs is proportional to sensor importance times radiance. The purpose of

this placement is to keep VPLs that are more important, i.e., they contribute more to the output image visible by the camera. The argument behind this placement strategy is that the VPLs should be positioned in the places that there is a lot of energy and they are visible by camera paths, either in the first sub-paths or farther. There is also an optional iterative relaxation step that can distribute VPLs in a scene, based on the idea proposed in photon relaxation paper [Spencer & Jones 2009].

2.3 Efficient Methods for Many-lights

Prior to Instant Radiosity, Ward [1994] improved rendering a scene with many light sources by pre-computing the shadows of the light sources. In this technique, the shadow casted by each light source is computed. Then, only light sources with strong enough shadows are considered for shading the scene. Shirley et al. [1996] proposed a new technique for defining a probability density function for computing direct illumination with only one sample, consequently reducing the rendering time.

Paquette et al. [1998] proposed a method using an octree to cluster the lights in a scene. In this octree, each node represents all the light sources it contains. In the shading pass, the error caused by rendering with these representative lights is calculated. A quality parameter based on the calculated error is computed such that it determines the path that should be taken in the hierarchy light structure. In each level, if the shading quality is not satisfactory, a lower level node in the hierarchy structure is chosen for the shading.

Local Illumination Environments was introduced [Fernandez et al. 2002] to speed up direct lighting in the scenes with many light sources by storing data related to occlusion,

geometric and radiometric into an octree. Each octree includes a set of lights and corresponding occluders in the scene. The lights in the octree are classified into three categories: fully visible, fully occluded, and partially occluded. Using this structure helps faster computation of direct illumination due to the fact that there is no need to send shadow rays for fully visible and fully occluded lights.

The Lightcuts method [Walter et al. 2005] is an important improvement for many-light methods that speeds up the rendering process a lot by taking a different approach. In Lightcuts, a special binary tree, called a light tree, is created for all point light sources in a scene (Figure 2.3). In this light tree, the lights in the scene are clustered together based on similarity in direction and intensity. The goal is to cluster similar lights in a way that they can be represented by a single light source, so that the number of light sources to be

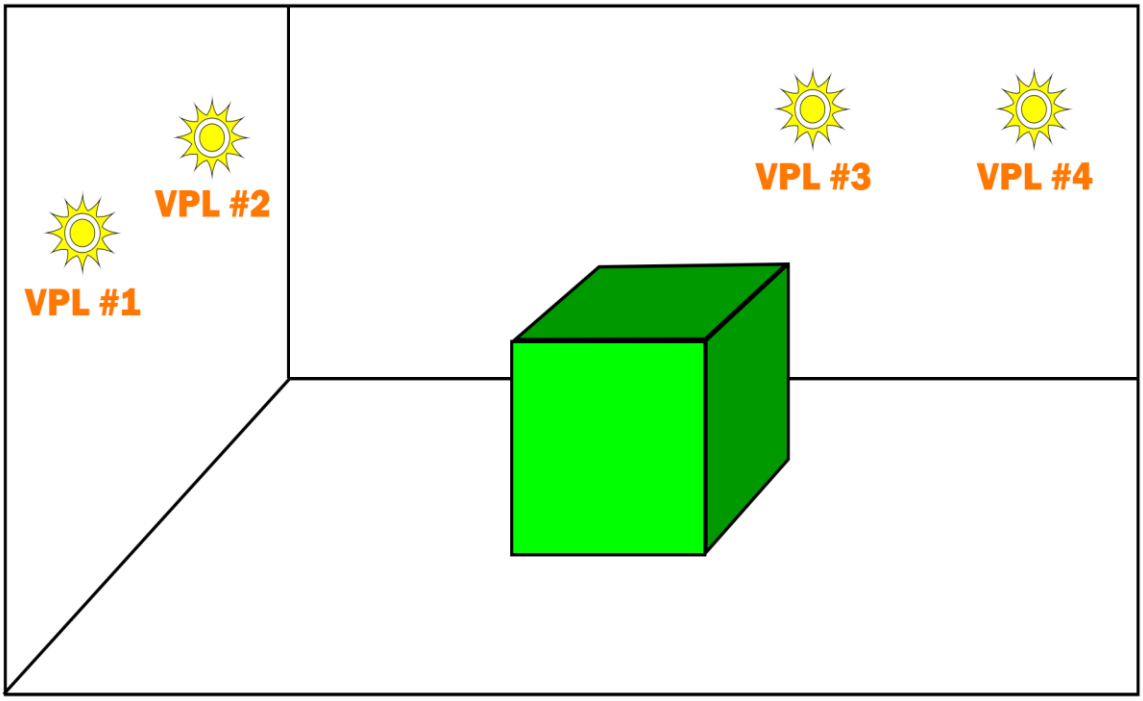


Figure 2.3: Sample scene

used in the final rendering step is reduced significantly. After creating a light tree for a scene, it can be used for shading any desired point in the scene; therefore, building a light tree is a one-time operation. At the end of this step, the root node of the light tree is one single light source which represents all light sources in the scene.

The radiance coming from direct illumination of a set of point lights can be written as

$$L_S(x,\omega) = \sum_{i \in S} M_i(x,\omega) G_i(x) V_i(x) I_i, \tag{2.6}$$

where S is a set of point lights, L is radiance cause by direct illumination at a surface point x, x is a surface point, ω is view direction, M_i is material term, G_i is geometric term, V_i is visibility term, and I_i is intensity.

Since the radiance should be calculated for every light source in a scene, the cost of computation is linear in number of light sources. To convert this linear computation to sub-linear, Lightcuts clusters similar lights together and specifies a representative light for each cluster

$$L_C(x,\omega) = \sum_{i \in C} M_i(x,\omega) G_i(x) V_i(x) I_i, \tag{2.7}$$

where C is a cluster, $C \subseteq S$, that is a set of point lights.

To compute the direct illumination from a cluster, we can use properties of the representative light to approximate the actual direct illumination of the cluster.

$$L_C(x,\omega) \approx M_j(x,\omega) G_j(x) V_j(x) \sum_{i \in C} I_i, \tag{2.8}$$

where $j \in C$ is the representing light of the cluster C.

The intensity of a cluster, $I_C = \sum I_i$, can be pre-computed and stored in the cluster. To form a light cluster properly, a light tree (Figure 2.4), which is binary tree, is created. In a light tree, leaf nodes are individual lights and the interior nodes are light clusters. Each light cluster is representing the lights that are stored in its child nodes. These child nodes can be either an individual light source or another light cluster.

Rendering a scene with a light cluster, instead of the actual light sources, can lead to artifacts in the final image. To minimize these artifacts, selecting a cluster is done in a way that the radiance estimation error, caused by the cluster selection, is so insignificant that the final rendered image is visually plausible to the viewer.

To render a scene, each point is illuminated by only a subset of light sources in the scene. This subset is called a light cut (Figure 2.5) and each representative light in the subset is guaranteed to contain less than a user-defined percentage (typically set as 2%) of the total illumination coming to a point. Consequently, for shading a point in a scene, only a small subset of light sources is used in direct illumination computation, which

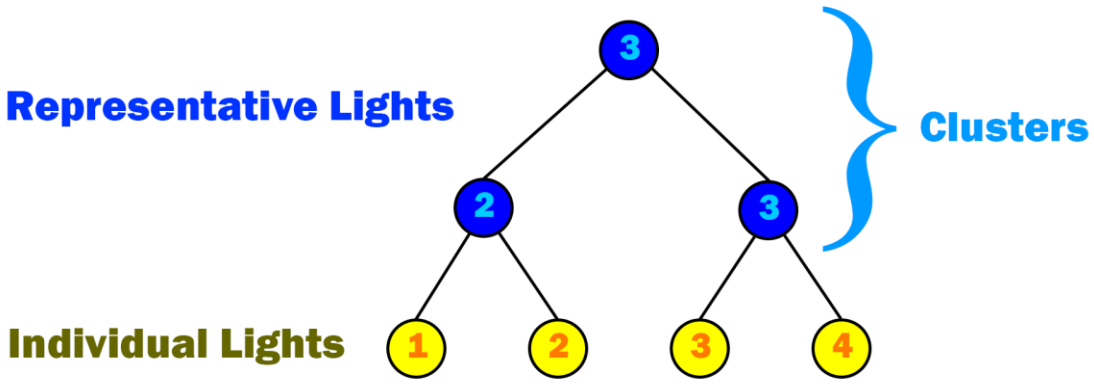


Figure 2.4: Light tree

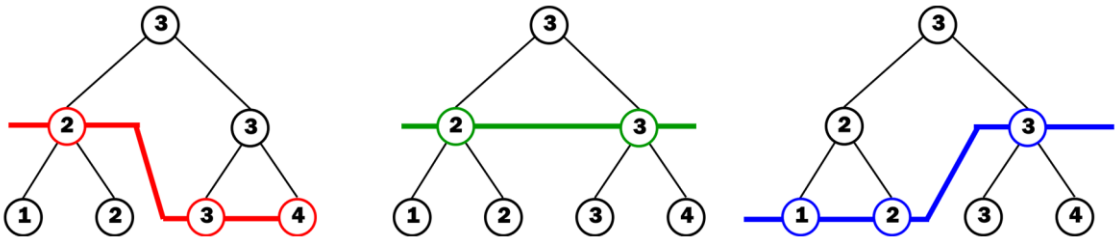


Figure 2.5: Three light cuts

results in a much faster rendering process.

In Lightcuts, the error bound calculation is very conservative and it is measured as a percentage of the light coming to the surface. Therefore, a light cut may include too many light sources in dark areas of a scene. To prevent this, there is limit for maximum number of light sources that can be in a light cut. If the limit is reached while computing the light cut, the point is shaded regardless of estimated error.

Hašan et al. [2007] proposed Matrix Row-Column Sampling (MRCS) which is similar to Lightcuts. Lightcuts computes a light cut for each point in the scene, but MRCS computes a single cut for the whole scene. There are some advantages and disadvantages to this approach. The first advantage is that the VPL importance can be done by computing shadow maps for the scene, which is faster if it is done in hardware level in GPU. The other advantage is that in MRCS, there is no need to find error bounds, because the cut is for the whole scene, not only one point in the scene. So, this information could be used throughout the process of rendering a scene, at least for one frame. The problem with MRCS is that because the cut is computed for some small number of VPL samples and then all other VPLs are mapped to these primary VPLs, it can produce some artifacts, especially for glossy surfaces.

LightSlice [Ou & Pellacini 2011] combines ideas of Lightcuts and MRCS to improve the computation of a light cut. Lightcuts re-computes a light cut for each point and this is computationally expensive, because this computation could be done multiple times. But for many of the points in a scene that are close to each other, the cut is the same. LightSlice finds these similar light cuts and uses the same computed light cut for the next points. In fact, LightSlice improves performance of the light cut computation by caching the computed light cuts for nearby points and using this data for the next light cut computation. Frederickx et al. [2015] extended LightSlice to work with virtual ray lights (VRLs). In this technique, the optimum number of light clusters is calculated adaptively.

Kollig & Keller [2006] proposed a method to avoid singularity caused by rendering a scene using VPLs. The method relies on a path tracing step to eliminate the artifacts caused by clamping the illumination of too-close VPLs. Engelhardt et al. [2012] introduced an approximate bias compensation to simulate participating media without any pre-computation. Novak et al. [2012] proposed a progressive algorithm to render indirect transport paths in volumetric media by generating virtual beam lights (VBLs) and avoiding the clamping too-close virtual lights. Huo et al. [2016] proposed a technique to efficiently gather the contributions of virtual lights in participating media. In this technique, only a small number of elements in “adaptive matrix column sampling” is considered to compute the final gathering of virtual lights.

2.4 Sample Elimination for Generating Poisson Disk Sample Sets

Due to the nature of random samples, randomly generated VPLs may be positioned in the places that are not useful for rendering the scene in the final pass. To achieve a higher

quality in the rendered image, we need to control where these VPLs are placed in a scene. We observed that VPLs that are too close to each other will waste the computational budget in the final rendering pass, and they can be represented by a single VPL if it is placed appropriately. Additionally, in some cases, there was no VPL in certain areas of the scene and this caused some artifacts in the final output image.

To prevent these situations, we need to uniformly distribute VPLs in the scene to increase the quality of the rendered image and to decease the rendering time simultaneously.

Yuksel [2015] proposed a method for generating Poisson disk samples via sample elimination that picks a subset of a sample set with the desired output size, rather than relying on a user-defined Poisson disk radius, such that the output subset has the blue noise property of Poisson disk sample sets. The advantage of specifying the subset size is that for many problems, it is more important to have a specific number of samples rather than to have samples that are farther than a specific radius.

To find the desired samples, a greedy algorithm is used that assigns a weight to each sample based on the distance to the neighbor samples, then removes the sample with the highest weight. After this sample removal, the weight of all samples around should be adjusted again.

To compute the weight w_i of a sample i, the weight contribution w_{ij} of all samples j is added up only if the sample j is within $2r_{max}$ distance of sample i, and $i \neq j$.

$$w_{ij} = \left(1 - \frac{\hat{d}_{ij}}{2r_{max}}\right)^{\alpha}, \qquad \hat{d}_{ij} = min(d_{ij}, 2r_{max}) \tag{2.9}$$

r_{max} value is different in various sampling domains, for example, $r_{max} = \sqrt{\frac{A_2}{2\sqrt{3}N}}$ in 2D and $r_{max} = \sqrt{\frac{A_3}{4\sqrt{2}N}}$ in 3D, where A_2 and A_3 are the area and volume of the sampling domain.

To achieve a better sample set, Yuksel advises that the initial sample set size should be 3 to 5 times greater than the desired size at the end. For example, if there is a need for 10K samples with Poisson disk sample properties, 50K samples should be generated and then, using this technique, 40K samples are eliminated to get the evenly distributed 10K samples at the end.

CHAPTER 3

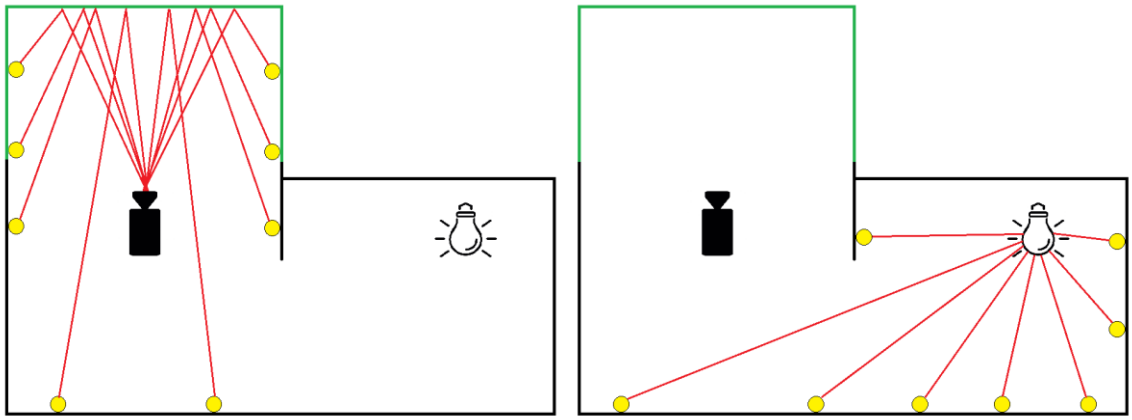
BLUE NOISE VPLS

3.1 Overview

Usually, many-light methods generate VPLs by starting from light sources, but in our algorithm, we start from the camera, because these VPLs contribute more to the final rendered image. The idea is that the illumination at the points visible by a camera comes directly from the VPLs visible by that point. Therefore, the best locations for VPLs are the secondary hit-points of camera rays, so that the VPLs can illuminate the points visible by the camera. This prevents placing a VPL on locations that, at the end, do not contribute to the final rendered image either due to occlusion from other objects in a scene, or because the outgoing illumination of the VPL is so low that it does not affect the color of a shaded surface point as much (Figure 3.1).

Briefly, to find the best VPL locations, firstly, a lot of VPLs are generated randomly and placed all over the scene. Then, many of these VPLs are eliminated using Poisson disk sample elimination technique. The result of this technique is that VPLs are distributed evenly in the scene and they are ready for the next step for radiance intensities computation.

To find the radiant intensity of VPLs, we do light tracing and to do that, we shoot a lot of photons from each light source in the scene proportional to their power. When a



(a) (b)

Figure 3.1: Placing VPLs starting (a) from the camera and (b) from the light source

photon hits a surface, we scatter the photon's energy to the VPLs near that hit-point.

The next step is to bounce the photon and repeat this process until either the photon does not hit anything or it reaches its maximum bounces.

At the end, we render the scene with the VPLs. To speed up the rendering step, we use Lightcuts, since Lightcuts computes the illumination for each point by using only a proper subset of VPLs in the scene, selected differently for each shaded point.

3.2 VPL Placement

Most algorithms that use VPLs for indirect illumination start from the light sources and trace light rays to find the position of VPLs in a scene. This make sense, since a VPL represents a light source and intuitively they should be traced from light sources. The problem with this approach is that in many cases, these VPLs do not contribute to the final image as much because they may not be reachable by the points that are visible by

camera (Figure 3.1 (b)).

In contrast, we start from camera to place VPLs in a scene. The reason why we do this is that the VPLs that are generated from camera paths are guaranteed to contribute to the output image. Also, we start placing VPLs in the scene after the first diffuse bounce, because the first diffuse hit-point is where we want to shade in the final rendering step and this hit-point could be illuminated by other VPLs in the scene (Figure 3.1 (a)).

3.3 VPL Generation and Elimination

Starting from the camera, a ray is generated and for each hit-point, we place a VPL at that point only if it is after the first diffuse bounce.

Notice that these VPLs are not actual virtual point lights, but merely placeholders. The term VPL might be confusing in this context because the VPLs are, in fact, point lights. But in this pass, our VPLs neither have radiant intensity nor have been generated from a light source. As a matter of fact, these VPLs are placeholders for the actual VPLs that we use in the last pass. For the moment, these VPLs store only surface normal and position of the hit-points.

We skip placing VPLs at the hit-points before a diffuse bounce because in the last pass, we shade this very first diffuse hit-point using the VPLs we generated. Therefore, there is no need to put a VPL at this point. In addition, it should be after a *diffuse* bounce because if the first hit is a specular bounce, we should trace the ray for the next bounce to approximate the illumination coming to that point.

The number of random VPLs generated in this step is 5 times larger than what we need for the rendering step, as advised by Yuksel [2015]. After generating all VPLs, we

eliminate most of them using the method that we described in section 2.4. Since the method works with specifying the number of output samples, we know exactly how many VPLs we remove from this set and how many VPLs remains in the scene. The resulting set of VPLs after this elimination, having more pronounced blue noise characteristics, is distributed evenly in the scene (Figure 3.2) and this helps generate an output image with higher quality.

3.4 VPL Radiant Intensity Estimation

To estimate the radiant intensity of VPLs generated in the previous step, we shoot lots of photons from each light source in a scene and trace them for any intersection with the objects in the scene. After a hit, we look around that hit-point in a specific radius to find VPLs in that area and scatter the energy of the photon to those VPLs. The photon's energy is distributed evenly among these VPLs, regardless of their distance to the hit-point. This may shift the energy distribution in the scene a little, but because the radius that we look for VPLs is very small, it does not cause any visual artifacts in the rendered

Figure 3.2: VPLs in Sponza scene. Middle: Rendered image. Left: 10K Random VPLs. Right: 10K blue noise VPLs (after elimination).

image.

Then, depending on the BRDF at the hit-point, the photon's energy is adjusted and the photon bounces to the next direction. This process is repeated until either the photon does not hit anything in the scene or it exceeds the maximum number of bounces. The whole photon shooting process is repeated until the maximum number of photons desired are shot for all lights in the scene.

No photon information is stored in this step and all the computation is done on the fly. At the end of this step, we have a collection of nicely distributed VPLs with computed radiant intensities that are a good estimation of light transport in the scene.

3.5 Final Rendering

To render a scene, we shoot camera rays into a scene and at each hit-point, we shade that point in two parts: First, we compute direct illumination coming from original light sources to that point. This is done by sending shadow rays to each light source and calculate the occlusion factor and radiance coming from each light source. Second, we shade the point using direct illumination from the VPLs we generated before. The direct illumination from these VPLs is, in fact, the estimation of indirect illumination for the aforementioned hit-point.

To compute the direct illumination from VPLs, we use Lightcuts, as we described in section 2.3. Using Lightcuts adds some error to the illumination computation of surfaces, but speeds up the rendering process significantly.

At the end, the output color of each point is the sum of direct illuminations from both original light sources and VPLs in the scene.

CHAPTER 4

RESULTS

For implementing our algorithm, we used Embree, a high-performance ray tracing framework developed by Intel, as the core of our code. We added our implementation of blue noise VPLs on top of Embree's path tracer.

All images in this chapter are rendered using a Windows 10 PC with an Intel Core i7-6700 3.4GHz Quad-Core Processor and 16GB of memory. To generate the reference images for each scene, we used the path tracer included in Embree.

For each scene, we assess the effectiveness of our technique in two areas. First, we show the differences in the output image of a scene generated by placing VPLs in the scene starting from the camera, compare to the output image of the scene generated by placing VPLs starting from the light sources. The second comparison is between rendering the scene with and without Poisson disk sampling for placing VPLs generated from the camera. For each scene, the reference image is shown before any comparison.

4.1 U-shaped Scene

In this scene (Figure 4.1), two rooms are connected by a narrow passage at the end. A light source is placed in the room on the right and we placed some objects, including a Utah teapot, in the room on the left (Figure 4.2). There is no light source in the left room,

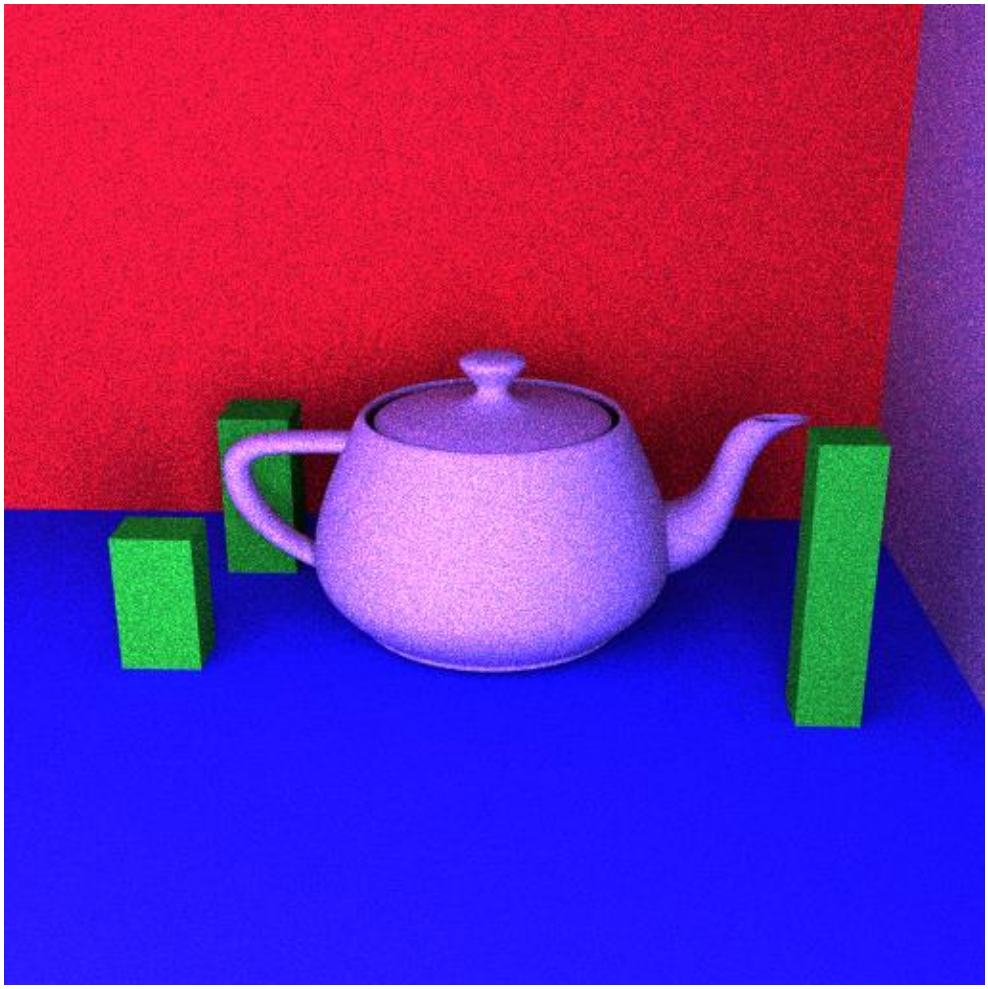


Figure 4.1: U-shaped scene rendered by path tracing.

so all illumination reaching the teapot (and other objects) comes from the light source in the right room after a couple of bounces off of the walls. Therefore, the only illumination visible to the camera is indirect illumination, reflected from the walls.

To render this scene, we generated 10M photons from the light source and 3K VPLs and rendered the scene using 8 samples per pixel. As it is shown in Figure 4.3, the output image is much dimmer when we generate VPLs from the light source (Figure 4.3 (b)). The reason is that the number of VPLs visible by the camera is very small because most of the VPLs are placed in the right room. However, if we use a very large number of

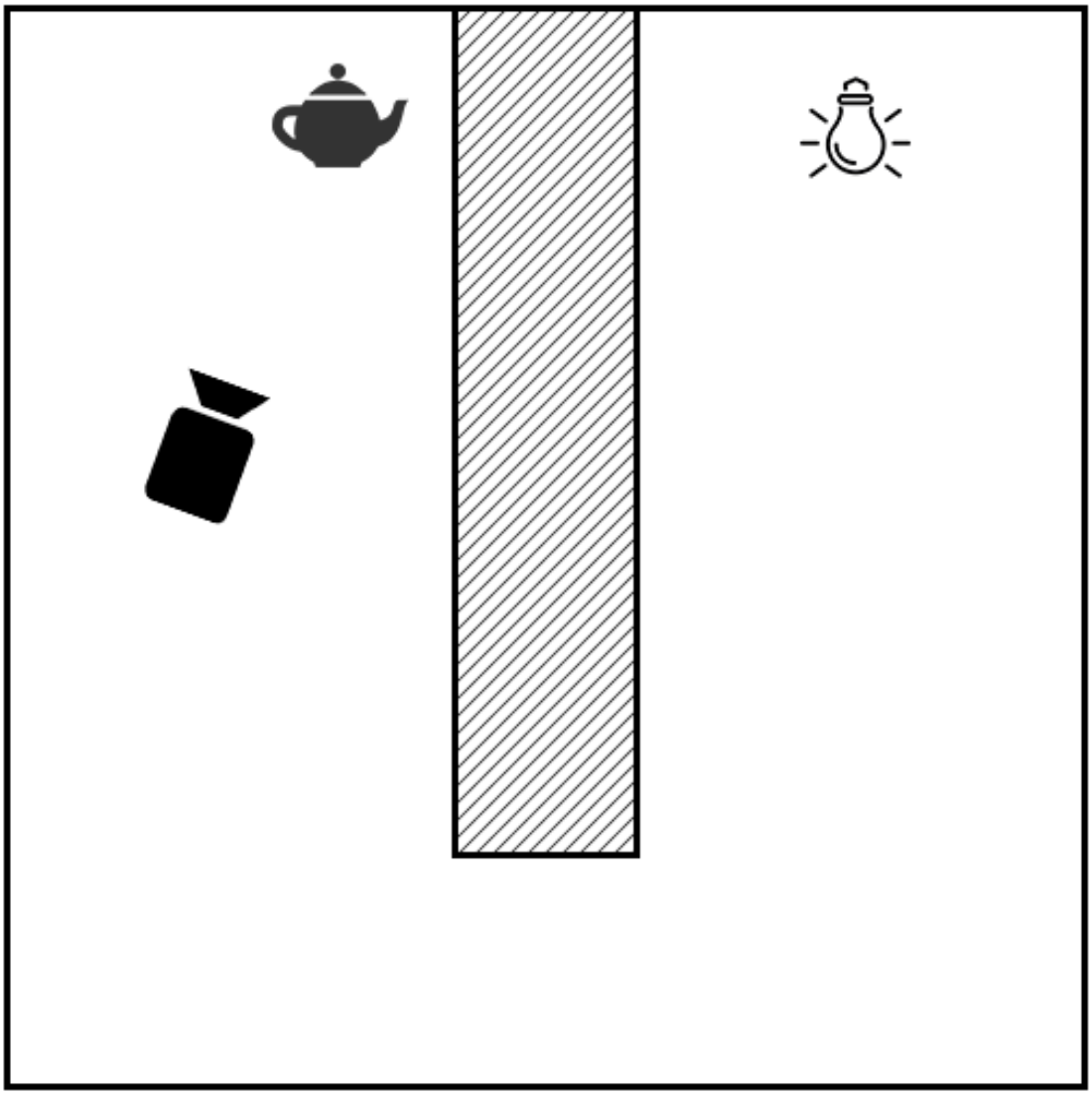
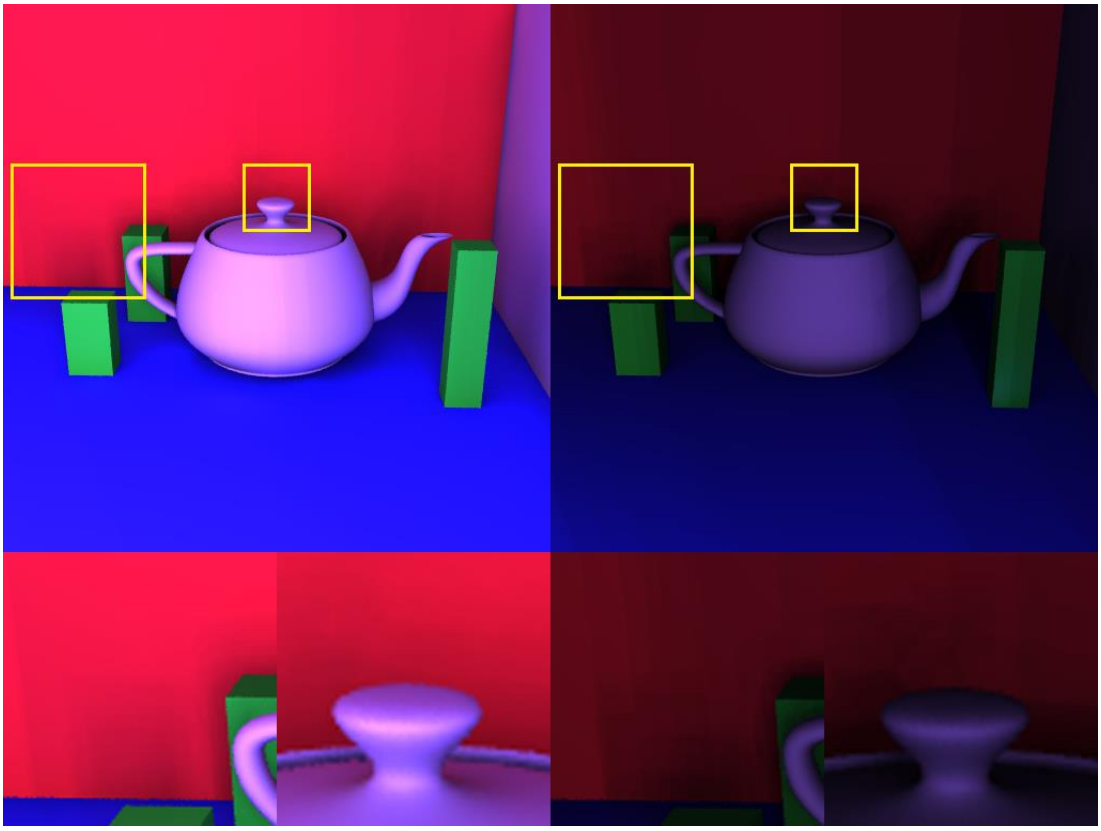


Figure 4.2: U-shaped scene from the top.

VPLs, both approaches converge to the same image.

Aside from the image brightness, there are some other differences in the shadows on the red wall. Shadows are smoother and artifacts are fewer in the image rendered by VPLs from the camera. These artifacts are common due to the nature of shading with VPLs.

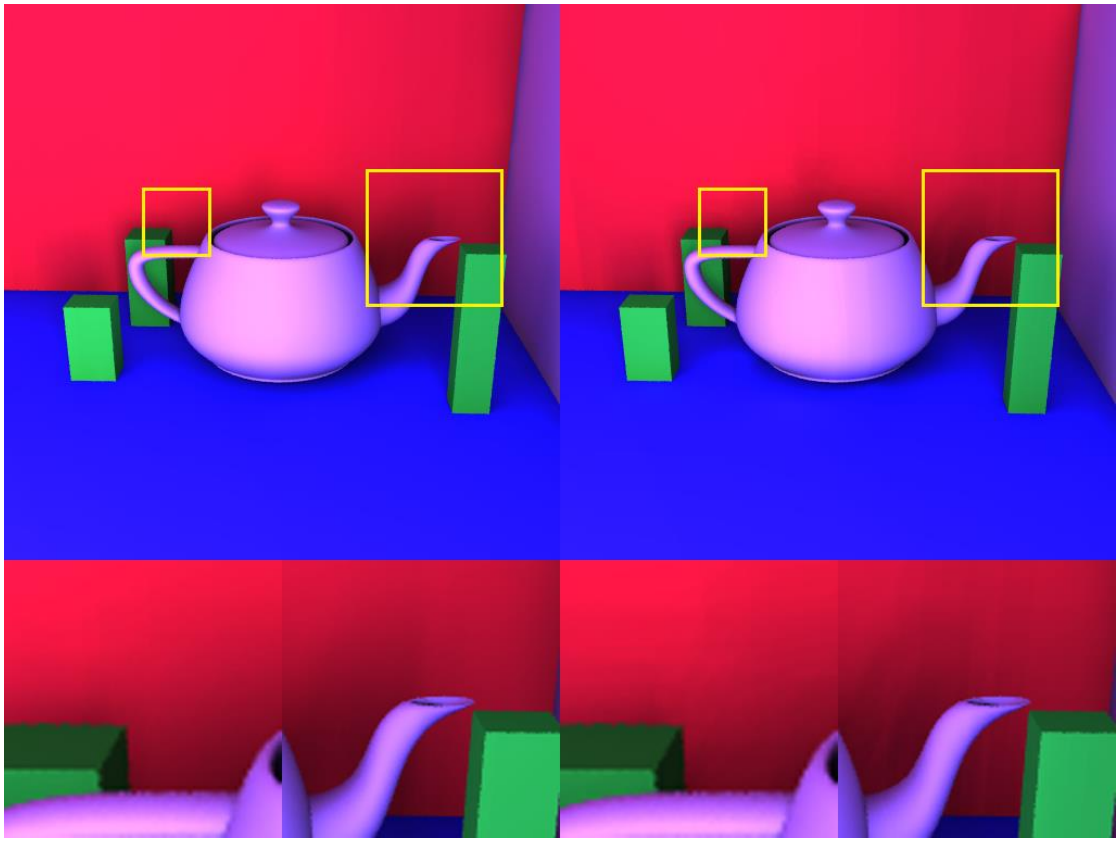


(a) (b)

Figure 4.3: Comparison of VPLs generated from (a) the camera and (b) the light source.

The image quality is higher in Figure 4.3(a) because a substantial portion of the VPLs generated from the camera contribute to the final image, as compared to the VPLs generated from the light source, most of which are not visible from the points visible to the camera.

The second comparison is for testing the effect of using Poisson disk samples in VPL distribution. Figure 4.4 shows the differences between randomly generated VPLs and blue noise VPLs, both generated from the camera.



(a) (b)

Figure 4.4: Comparison of (a) blue noise VPLs and (b) random VPLs.

The main differences between the two images are the artifacts in the teapot's shadow. In Figure 4.4(b), a number of distinct shadows are visible, while in Figure 4.4(a), the shadows appear soft. Also, there is a spot on the red wall where the shadow of the green box is shown. In Figure 4.4(b), the shadow consists of a couple of dark areas, but the same spot is correctly rendered in Figure 4.4(a). Because the VPLs in Figure 4.4(a) are uniformly distributed, the rendered scene contains fewer artifacts.

4.2 Veach Door Scene

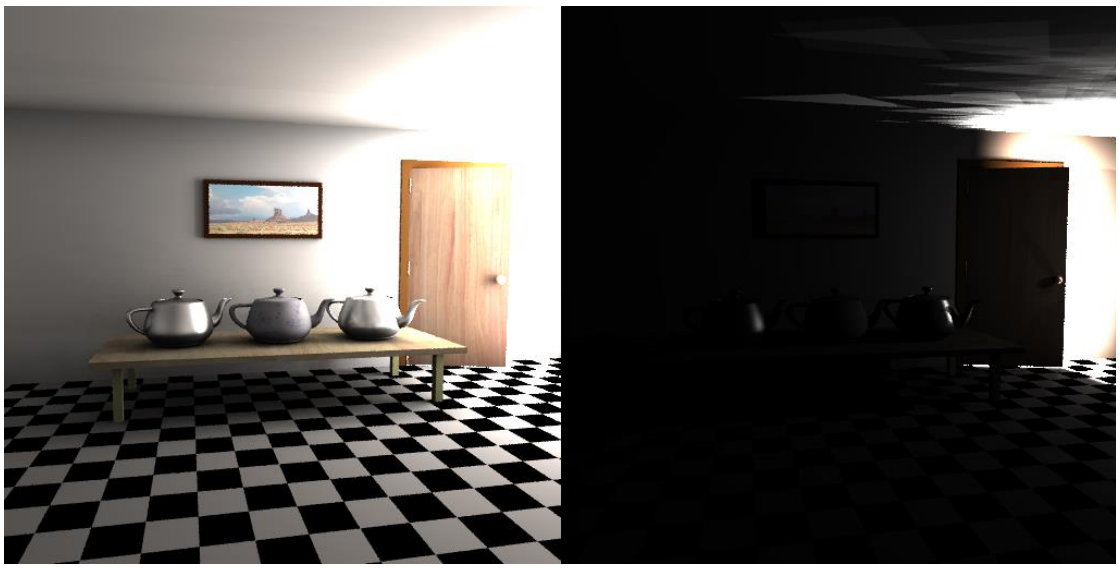
A recreation of the famous Veach door scene [Veach & Guibas 1997] includes a door that is slightly open and is the only light source is in the room behind that door (Figure 4.5). Only a small percentage of the light in the other room (containing the light source) goes through the door and illuminates the room, where the camera is.

To render this scene using blue noise VPLs, we generated 6K VPLs and 10M photons

Figure 4.5: Veach door scene rendered by path tracing.

and rendered the scene using 8 samples per pixel. We set up the same scenarios, as we did for the previous scene, to compare the results. Figure 4.6 demonstrates the output images of the scenes rendered by VPLs from the camera and VPLs from the light source.

As expected, when we rendered the scene using VPLs from the light source, the output image is almost completely dark. There are only some spots near the door that were illuminated due to the fact that in Figure 4.6(b), only a small portion of VPLs are positioned in the room visible by the camera. When the VPLs are generated from the light source, this scene requires much more VPLs for approximating the lighting in the room with the camera. In comparison, when the VPLs are generated from the camera, we can capture the illumination by tracing a large number of photons for determining the intensities of those VPLs, as shown in Figure 4.6(a).

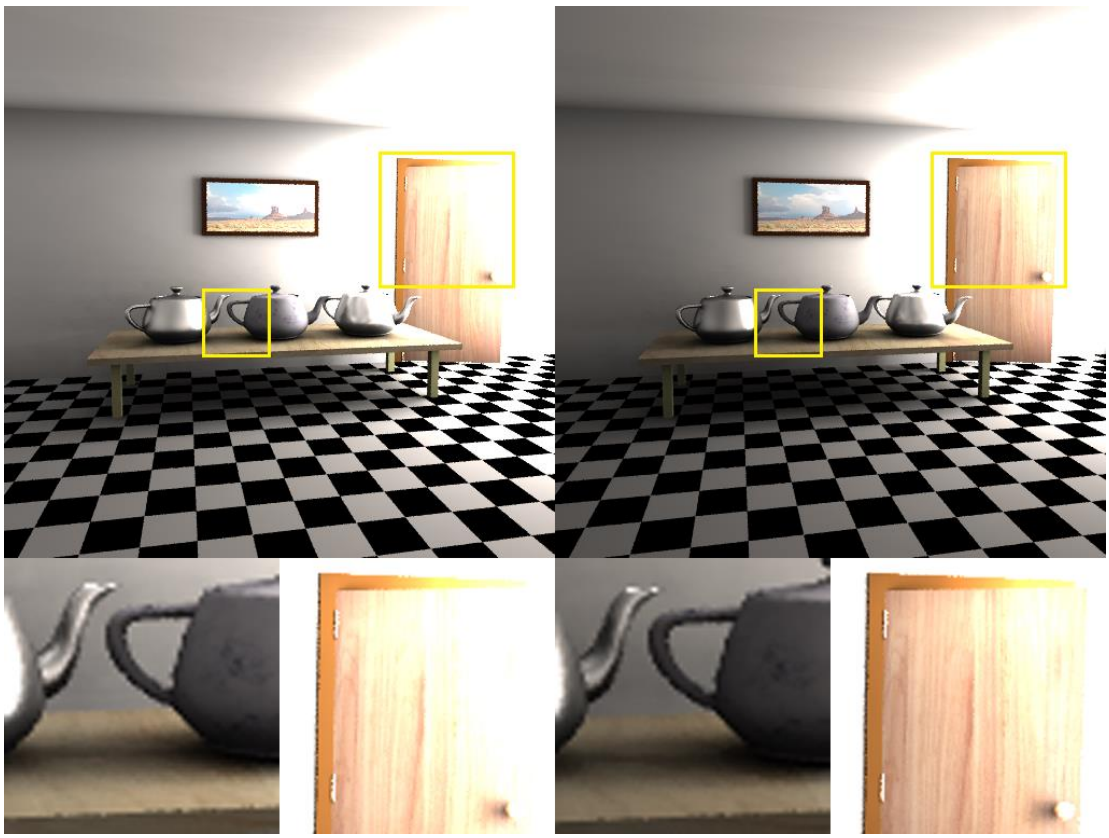


(a) (b)

Figure 4.6: Comparison of VPLs generated from (a) the camera and (b) the light source.

In the next comparison, we rendered the scene with and without Poisson disk sampling for placing the VPLs generated from the camera (Figure 4.7).

Like the previous scene, there are more artifacts in the Figure 4.7(b), especially in the shadow casted by the teapot on the table. In addition, the illumination of bright spots on top of the door is more accurate in the image rendered using VPLs generated by Poisson disk sampling technique.



(a) (b)

Figure 4.7: Comparison of (a) blue noise VPLs and (b) random VPLs.

4.3 Sponza Scene

To render the Sponza scene (Figure 4.8) [McGuire 2011], we generated 3K VPLs and 10M photons and rendered the scene using 8 samples per pixel. Since the number of VPLs is small, the image rendered by VPLs from camera is brighter than the image rendered by VPLs from the light source (Figure 4.9), because many of the VPLs generated from the light source are positioned in the places that do not contribute to the rendered image.

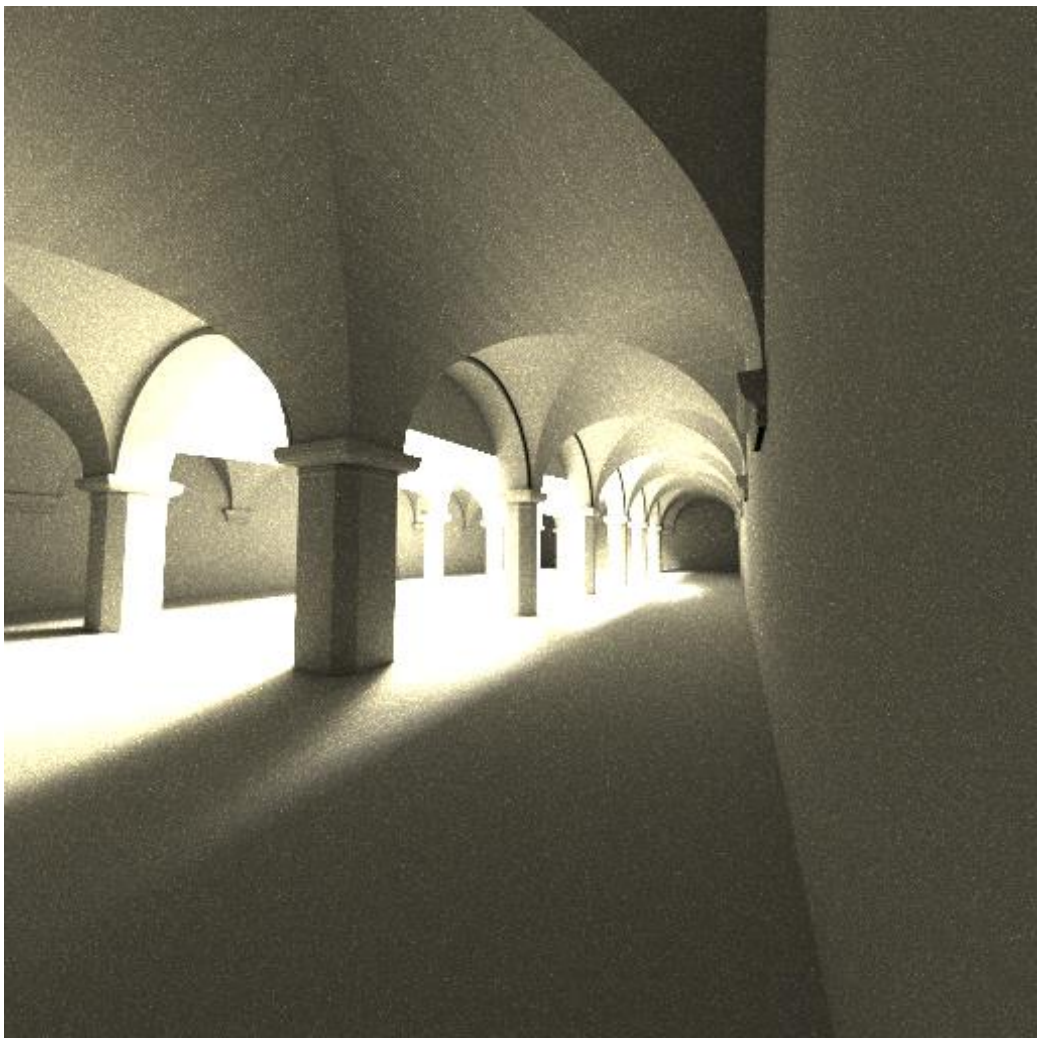
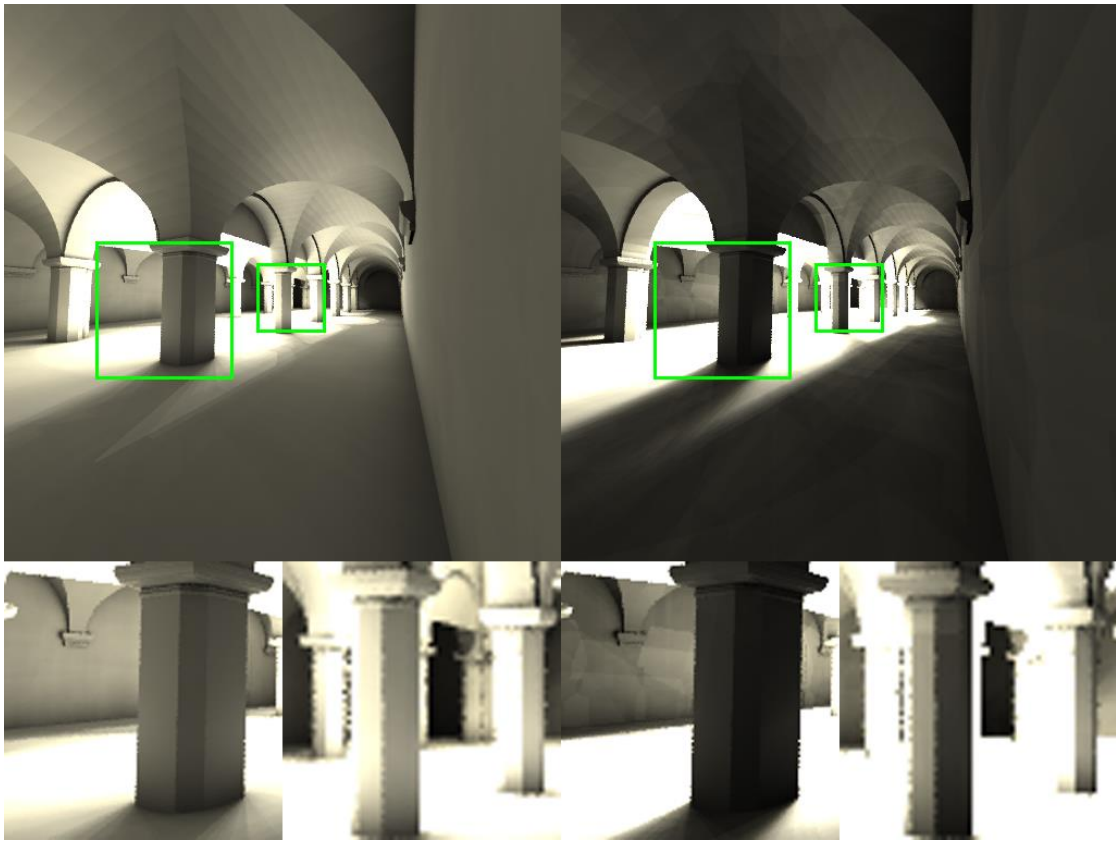


Figure 4.8: Sponza scene rendered by path tracing.

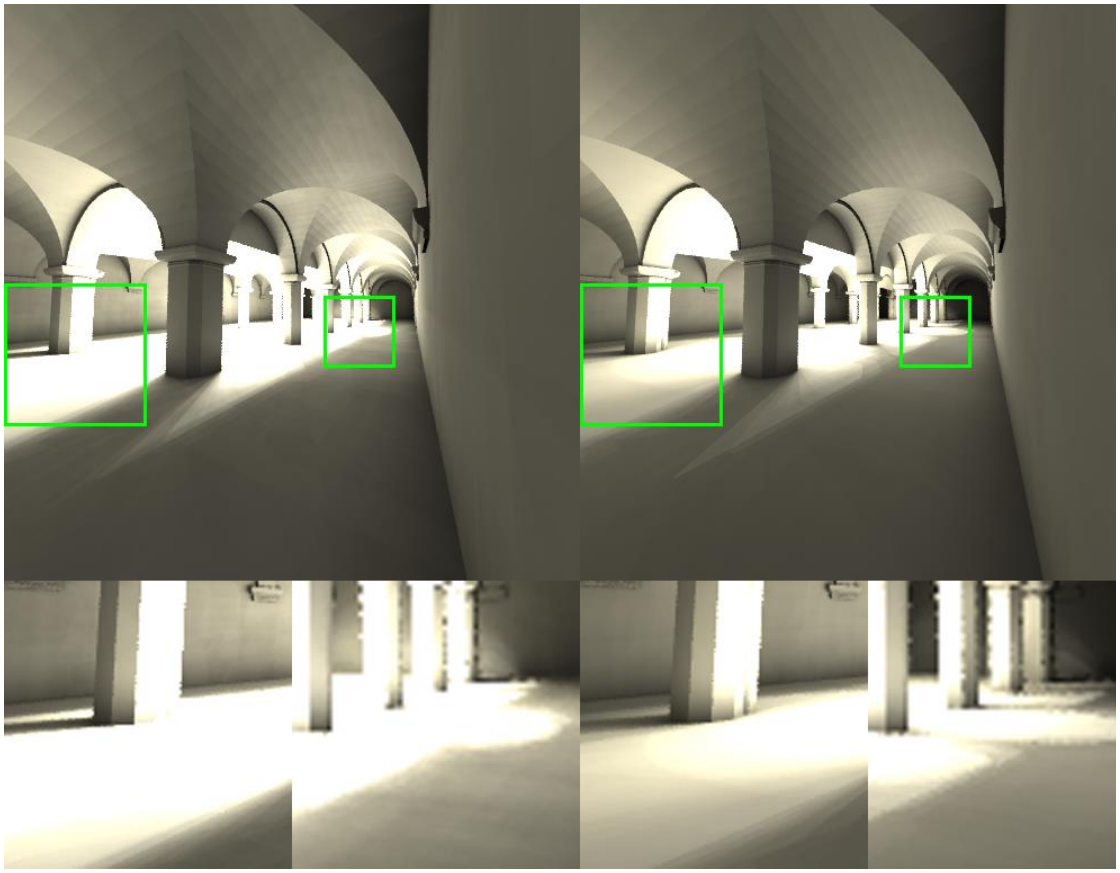


(a) (b)

Figure 4.9: Comparison of VPLs generated from (a) the camera and (b) the light source.

As it is shown, the quality of the image in Figure 4.9(a) is higher because we rendered the scene using VPLs generated from the camera. Besides, the illumination of the scene is more plausible.

In Figure 4.10, the image rendered by blue noise VPLs has higher quality in the bright spot on the left side of the scene. Since the VPLs are evenly distributed in the scene, the output image is more realistic (Figure 4.10 (a)) compared to the image rendered by randomly generated VPLs (Figure 4.10 (b)).



(a) (b)

Figure 4.10: Comparison of (a) blue noise VPLs and (b) random VPLs.

CHAPTER 5

CONCLUSION

Blue noise VPLs is a new technique for improving the quality of rendered images using VPLs. Generating VPLs from a camera causes the VPLs to contribute better to the illumination of the points visible by a camera. Furthermore, Poisson disk sampling results in a uniformly distributed set of VPLs in the part of the scene that is indirectly visible to the camera, which consequently generates an output image with fewer visible artifacts.

Thus, the tests included in this thesis demonstrate that careful placement of VPLs in the scene such that the VPLs are neither too-close-to nor too-far-from each other results in improved rendering quality. There are some situations in which the quality differences are more noticeable, especially in the scene that the light's frustum does not intersect with the camera frustum as much.

REFERENCES

Cohen, M et al. 1993, *Radiosity and Realistic Image Synthesis*.

Dachsbacher, C et al. 2014, 'Scalable Realistic Rendering with Many-Light Methods', *Computer Graphics Forum*, vol 33, pp. 88-104.

Dachsbacher, C & Stamminger, M 2005, 'Reflective Shadow Maps', *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA.

Davidovič, T et al. 2012, 'Progressive Lightcuts for GPU', *ACM SIGGRAPH 2012 Talks*, ACM, New York, NY, USA.

Davidovič, T et al. 2010, 'Combining Global and Local Virtual Lights for Detailed Glossy Illumination', *ACM Trans. Graph.*, vol 29, pp. 143:1--143:8.

Engelhardt, T et al. 2012, 'Approximate Bias Compensation for Rendering Scenes with Heterogeneous Participating Media', *Computer Graphics Forum (Proceedings of Pacific Graphics 2012)*, vol 31, pp. 2145-2154.

Fernandez, S et al. 2002, 'Local Illumination Environments for Direct Lighting Acceleration', *Proceedings of the 13th Eurographics Workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Frederickx, R et al. 2015, 'Adaptive LightSlice for Virtual Ray Lights', *EG 2015 -- Short Papers*, The Eurographics Association.

Georgiev, I & Slusallek, P 2010, 'Simple and Robust Iterative Importance Sampling of Virtual Point Lights', *Eurographics 2010 - Short Papers*, The Eurographics Association.

Hašan, M et al. 2009, 'Virtual Spherical Lights for Many-light Rendering of Glossy Scenes', *ACM Trans. Graph.*, vol 28, pp. 143:1--143:6.

Hašan, M et al. 2007, 'Matrix Row-column Sampling for the Many-light Problem', *ACM SIGGRAPH 2007 Papers*, ACM, New York, NY, USA.

Hedman, P et al. 2016, 'Sequential Monte Carlo Instant Radiosity', *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA.

Huo, Y et al. 2016, 'Adaptive Matrix Column Sampling and Completion for Rendering Participating Media', *ACM Trans. Graph.*, vol 35, pp. 167:1--167:11.

Jensen, HW & Christensen, NJ 1995, 'Photon maps in bidirectional Monte Carlo ray tracing of complex objects', *Computers \& Graphics*, vol 19, pp. 215-224.

Kajiya, JT 1986, 'The Rendering Equation', *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA.

Keller, A 1997, 'Instant Radiosity', *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Kollig, T & Keller, A 2006, 'Illumination in the Presence of Weak Singularities', in H Niederreiter, D Talay (eds.), *Monte Carlo and Quasi-Monte Carlo Methods 2004*, Springer Berlin Heidelberg, Berlin, Heidelberg.

McGuire, M 2011, *Computer Graphics Archive*, small http://graphics.cs.williams.edu/data.

Novak, J et al. 2012, 'Progressive Virtual Beam Lights', *Comput. Graph. Forum*, vol 31, pp. 1407-1413.

Novák, J et al. 2012, 'Virtual Ray Lights for Rendering Scenes with Participating Media', *ACM Trans. Graph.*, vol 31, pp. 60:1--60:11.

Ou, J & Pellacini, F 2011, 'LightSlice: Matrix Slice Sampling for the Many-lights Problem', *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA.

Paquette, E et al. 1998, 'A Light Hierarchy for Fast Rendering of Scenes with Many Lights', *Computer Graphics Forum*, vol 17, pp. 63-74.

Raab, M et al. 2008, 'Unbiased Global Illumination with Participating Media', in A Keller et al. (eds.), *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Segovia, B et al. 2006, 'Bidirectional Instant Radiosity', *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Segovia, B et al. 2007, 'Metropolis Instant Radiosity', *Computer Graphics Forum*, vol 26, pp. 425-434.

Shirley, P et al. 1996, 'Monte Carlo Techniques for Direct Lighting Calculations', *ACM Trans. Graph.*, vol 15, pp. 1-36.

Simon, F et al. 2015, 'Rich-VPLs for Improving the Versatility of Many-Light Methods', *Comput. Graph. Forum*, vol 34, pp. 575-584.

Spencer, B & Jones, MW 2009, 'Into the Blue: Better Caustics through Photon Relaxation', *Computer Graphics Forum*, vol 28, pp. 319-328.

Veach, E & Guibas, L 1995, 'Bidirectional Estimators for Light Transport', in G Sakas et al. (eds.), *Photorealistic Rendering Techniques*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Veach, E & Guibas, LJ 1997, 'Metropolis Light Transport', *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Wald, I et al. 2003, 'Interactive Global Illumination in Complex and Highly Occluded Environments', *Proceedings of the 14th Eurographics Workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Wald, I et al. 2002, 'Interactive Global Illumination Using Fast Ray Tracing', *Proceedings of the 13th Eurographics Workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

Walter, B et al. 2006, 'Multidimensional Lightcuts', *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA.

Walter, B et al. 2005, 'Lightcuts: A Scalable Approach to Illumination', *ACM Trans. Graph.*, vol 24, pp. 1098-1107.

Walter, B et al. 2012, 'Bidirectional Lightcuts', *ACM Trans. Graph.*, vol 31, pp. 59:1--59:11.

Wang, R et al. 2013, 'GPU-based Out-of-core Many-lights Rendering', *ACM Trans. Graph.*, vol 32, pp. 210:1--210:10.

Ward, GJ 1994, 'Adaptive Shadow Testing for Ray Tracing', in P Brunet, FW Jansen (eds.), *Photorealistic Rendering in Computer Graphics: Proceedings of the Second Eurographics Workshop on Rendering*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Yuksel, C 2015, 'Sample Elimination for Generating Poisson Disk Sample Sets', *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, vol 34, pp. 25-32.