國立臺灣大學電機資訊學院資訊工程學系暨研究所
博士論文
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Dissertation

使用取樣與重建技術的高效率蒙地卡羅渲染法
Sampling and Reconstruction Techniques for Efficient Monte Carlo Rendering

吳昱霆
Yu-Ting Wu

指導教授：莊永裕 博士
Advisor: Yung-Yu Chuang, Ph.D.

中華民國 103 年 6 月
June, 2014

中文摘要

產生視覺逼眞的影像以及進行數值精確的科學模擬爲電腦圖學技術的兩個重要目的，基於物理性質的渲染繪製 (physically-based rendering) 因爲可以同時滿足這兩個目的而受到重視。爲了能夠使用一套完整的理論來考慮各式各樣複雜的光線傳播路徑，其通常會採用蒙地卡羅光線追蹤法 (Monte Carlo ray tracing) 來進行模擬。雖然蒙地卡羅法能用簡潔的方式模擬多種光影效果， 其收斂速度並不理想。當繪製一個擁有大量的三維模型、逼眞的材質、 以及複雜光源的場景時，蒙地卡羅光線追蹤法往往需要非常大量的取樣數才能才能得到一張沒有雜訊的影像。

在這篇博士論文中我們共提出三種取樣與重建技術來增進蒙地卡羅光線追蹤法的效率。 首先，我們觀察到在繪製複雜的場景時，比起光源和材質，物體表面兩點的可視度 (visibility) 計算通常是造成雜訊的主要原因。因此爲了增加取樣的效率，我們提出了可適度群聚 (VisibilityCluster) 演算法來快速並準確地預測場景中的可視度。此演算法可以被整合進重要性取樣 (importance sampling) 的架構中並大量減少雜訊。 接著爲了加速繪製需要龐大計算量的半透明材質，我們首先提出一個雙重矩陣(Dual-matrix)表示法來詮釋渲染半透明材質的問題。同時，我們發展一套取樣與重建技術來減少不重要的計算，大幅加速半透明材質的繪製。 最後，我們提出一個有效率的自適應取樣與重建 (adaptive sampling and reconstruction) 架構來處理淺景深，動態模糊，

以及全域照明等光線傳遞效果。藉著導入統計學中估計誤差的方法到三維繪製的領域，我們成功的決定每個像素需要的取樣數量，並估計出一個最好的濾波器 (filter) 來大幅減少雜訊。 與之前的研究相比，以上的這三個方法皆能大幅改善蒙地卡羅光線追蹤法的繪製效率。

Abstract

Two of the most important tasks that computer graphics techniques try to solve is rendering photo-realistic images and performing numerically accurate simulation. Physically-based rendering can naturally satisfy these two goals. It is usually simulated by the Monte Carlo ray tracing for handling a variety of sophisticated light transport paths in a united manner. Despite its generality and simplicity, however, Monte Carlo integration converges slowly. Rendering scenes with lots of complex geometry and realistic materials under complex illumination usually requires a large number of samples to produce a noise-free image.

In this dissertation, we proposed three advanced sampling and reconstruction algorithms for improving the performance of Monte Carlo integration. First, realizing that in complex scenes visibility is usually the major source of noise during sampling the shading function, we developed a method called VisibilityCluster for efficiently approximating visibility function. By integrating it into importance sampling framework, we achieve superior noise reduction compared to previous approaches. Second, to reduce the computation overhead of rendering translucent materials, we proposed an algorithm, Dual-matrix sampling, to avoid evaluating unimportant surface samples which contribute little to the final image. Finally, a general adaptive sampling and reconstruction framework named SURE-based optimization is proposed to ren-

der a wide range of distributed effects, including depth of field, motion blur, and global illumination. All of the three methods achieve significant performance improvement compared to the state-of-the-art rendering algorithms.

Table of Contents

Abstract **v**

List of Figures **x**

Chapter 1 Introduction **1**

Chapter 2 Light Transport, Monte Carlo, and Path Integration **8**

2.1 Light Transport 8

2.2 Monte Carlo Integration 9

2.3 Importance Sampling 12

2.4 Path Integration 13

2.4.1 Path integral formulation 13

2.4.2 Antialiasing, depth of field, and motion blur 16

2.5 Path-based Rendering Algorithms 17

2.5.1 Path tracing 17

2.5.2 Virtual point light 18

Chapter 3 VisibilityCluster: Average Directional Visibility for Many-Light Rendering **20**

3.1 Overview 21

3.2 Related Work 24
3.2.1 Importance sampling 24
3.2.2 Visibility algorithms 26
3.2.3 Directional occlusion 27
3.3 Background 28
3.4 VisibilityCluster 29
3.4.1 Initial clustering 30
3.4.2 Average visibility estimation 32
3.4.3 Local refinement 32
3.4.4 Bias avoidance 34
3.5 Experiments 35
3.5.1 Triple-product importance sampling 36
3.5.2 Directional occlusion 44
3.6 Conclusion and Future Work 46

Chapter 4 Dual-Matrix Sampling for Scalable Translucent Material Rendering 48

4.1 Overview 49
4.2 Related Work 52
4.2.1 BSSRDF models for subsurface scattering 52
4.2.2 Subsurface scattering rendering 53
4.3 Dual-Matrix Representation and Sampling 55
4.3.1 Algorithm Overview 55
4.3.2 Dual-matrix Sampling 57
4.4 Experiments 61
4.5 Conclusion and Future Work 68

Chapter 5 SURE-based Optimization for Adaptive Sampling and Reconstruction 71

5.1 Overview 72

5.2 Related Work 74

5.2.1 Adaptive sampling and reconstruction 74

5.2.2 Denoising using SURE 76

5.3 Stein's Unbiased Risk Estimator (SURE) 76

5.4 SURE-based Adaptive Rendering 77

5.4.1 Initial samples 79

5.4.2 Filter selection using SURE 79

5.4.3 Adaptive sampling 84

5.5 Experiments 84

5.5.1 Parameter setting 85

5.5.2 Comparisons 86

5.5.3 Discussions 90

5.5.4 Other filters 92

5.5.5 Limitations 95

5.6 Conclusion and Future Work 95

Chapter 6 Conclusions 97

Chapter A Derivatives for Filters 1

Bibliography 2

List of Figures

1.1 Movies rendered with Arnold, a physically-based renderer developed by Solid Angle. 2

1.2 Examples of light transport paths in real-world photos. 3

1.3 One example that Monte Carlo methods tend to produce noisy result when the number of samples is not enough. 4

1.4 Statistics for the total rendering hours one DreamWork's feature animation took. 5

3.1 Examples for limitations of previous importance sampling methods. . . . 21

3.2 Visualization of visibility matrix 23

3.3 The flowchart for constructing VisibilityClusters. 29

3.4 Visualization of light clustering and shading point clustering. 30

3.5 Average visibility estimation. 32

3.6 Comparisons of VisibilityClusters with and without local refinement. . . . 34

3.7 The reason why we only split light clusters when constructing VisibilityClusters. 35

3.8 The experiment for determining how many shadow rays should be used in average visibility estimation. 37

3.9 Equal-time comparison (150 sec.) of the Town scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 38

3.10 Equal-time comparison (200 sec.) of the Lunch scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 38

3.11 Equal-time comparison (300 sec.) of the Killeroo scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 39

3.12 Equal-time comparison (600 sec.) of the Sponza scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 41

3.13 Equal-time comparison (300 sec.) of the Room scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 41

3.14 Equal-time comparison (500 sec.) of the Construction scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 42

3.15 Equal-time comparison (800 sec.) of the Hairball scene between bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 42

3.16 Error visualization of the seven scenes rendered by bidirectional importance sampling, irradiance caching, and our VisibilityCluster. 43

3.17 Directional occlusion with VisibilityCluster. 45

4.1 Several scenes which are inefficient to render with previous methods for translucent material rendering. 49

4.2 Matrix visualization of Light-to-Surface matrix and Surface-to-Camera matrix. 50

4.3 Flowchart of the dual-matrix sampling algorithm. 56

4.4 Algorithm for Rd estimation. 58

4.5 Equal-time comparison of the refining metrics: maximum solid angle and our approach. 62

4.6 Equal-time comparison of the Sculpture scene (30 sec.) between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling. 63

4.7 Equal-time comparison of the Cathedral scene (30 sec.) between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling. 64

4.8 Equal-time comparison of the Room scene (300 sec.) between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling. . . 65

4.9 Equal-time comparison of the ChessGame scene (110 sec.) between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling. 65

4.10 Equal-time comparison of the Exhibition scene (300 sec.) between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling. 66

4.11 Equal-quality comparisons on the *Sculpture*, textitCathedral, and *Exhibition* scenes between traditional two-pass method, subsurface lightcuts, and our dual-matrix sampling method. 70

5.1 Comparisons between greedy error minimization and our SURE-based filtering. 72

5.2 The flowchart of our SURE-based adaptive rendering framework. 78

5.3 Comparisons of filtering the Sibenik scene with depth-of-field effects using L2 distance and our normalized distance. 81

5.4 Visualization of the scale selection map for σ_s of our method. 83

5.5 Visualizations for the sampling density of our approach. 85

5.6 Comparisons on the Sibenik scene between standard Monte Carlo path tracing, greedy error minimization, random parameter filtering, and our SURE-based adaptive rendering. 87

5.7 Comparisons on the Teapot scene between standard Monte Carlo path tracing, greedy error minimization, random parameter filtering, and our SURE-based adaptive rendering. 88

5.8 Comparisons on the Sponza scene between standard Monte Carlo path tracing, greedy error minimization, random parameter filtering, and our SURE-based adaptive rendering. 89

5.9 Comparisons on the Town scene between standard Monte Carlo path tracing, greedy error minimization, random parameter filtering, and our SURE-based adaptive rendering. 90

5.10 Error visualization of the four scenes rendered by standard Monte Carlo path tracing, greedy error minimization, random parameter filtering, and our SURE-based adaptive rendering. 91

5.11 Comparisons between greedy error minimization and our SURE-based adaptive rendering on using isotropic filters only. 93

5.12 Comparison of cross non-local means filters without and with SURE-based framework. 94

List of Tables

2.1 Examples of bad distributions could hurt variance. 13

3.1 Scene profiles and rendering settings for the seven compared scenes. . . . 36

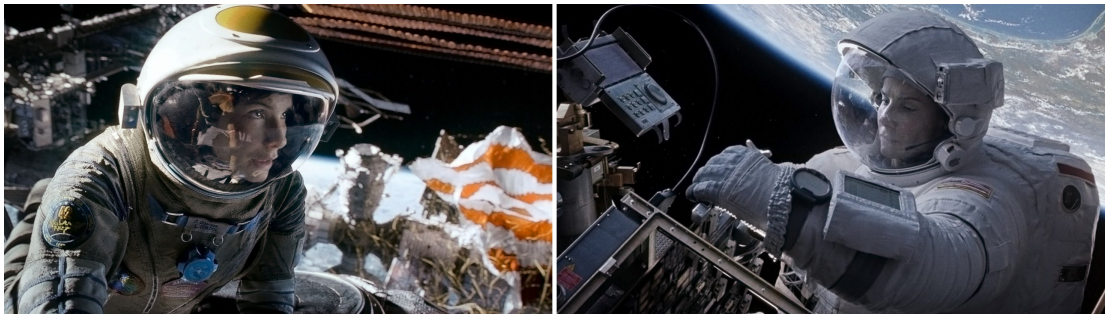
4.1 Statistics of the five test scenes. 64

Chapter 1

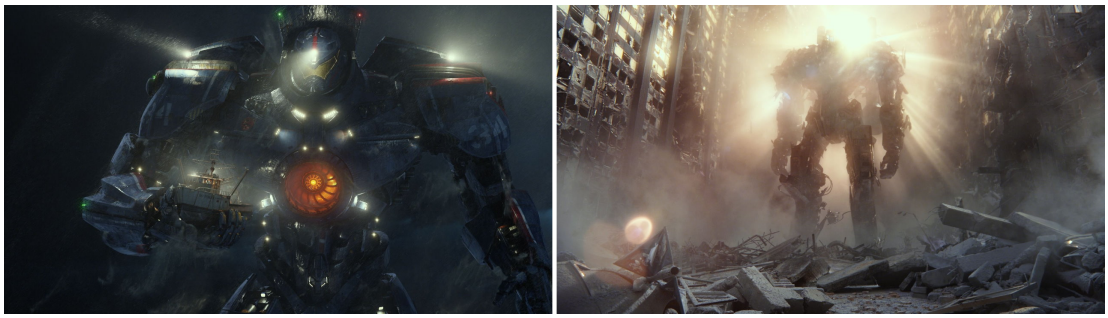
Introduction

After over thirty years of research, computer graphics techniques becomes mature and have been widespread in our daily life. Nowadays, we can find that computer generated images are widely used in newspapers, magazines, and journalism. Lifelike virtual 3D models such as dinosaurs and monsters can be seamlessly composited into movies and television programs. For some people, 3D animations and video games are their major entertainments. Obviously, the development of computer graphics has brought a revolution of media contents.

One of the most important tasks that computer graphics techniques try to solve is to reproduce the appearances of real-world objects. This is usually related to generate physically correct and photo-realistic images. For examples, in the applications of lighting and architecture design and visualization of scientific and medical data, we require the appearances of the rendered objects to be as close as possible to their real looks. We also need physically correct results to avoid wrong judgement. To fulfill these requirements, physically-based rendering, which simulates the interactions between lights, virtual shapes, and realistic materials based on physics and mathematics, becomes an extremely important research topic. In addition, because physically-based rendering can naturally generate photo-realistic images, it has also been widely used in recent film pro-



(a) Gravity



(b) Pacific Rim

(c) Iron Man 3

Figure 1.1: Movies rendered with Arnold, a physically-based renderer developed by Solid Angle. Upper: Gravity; Middle: Pacific Rim; Bottom: Iron Man 3. Images from the gallery of Solid Angle.

duction to offload the artists. Figure 1.1 shows several feature movies which are rendered using the Arnold, a physically-based renderer developed by Solid Angle.

To successfully generate images which are indistinguishable from photographs, a variety of sophisticated light transport paths, including reflection 1.2(a), refraction 1.2(b), global illumination 1.2(c), and subsurface scattering 1.2(d), should be simulated. Effects

(a) Reflection

(b) Refraction

(c) Global illumination

(d) Subsurface scattering

(e) Depth of field

(f) Motion blur

Figure 1.2: Examples of light transport paths in real-world photos.

produced by the lens system of cameras like depth of field 1.2(e) and motion blur 1.2(f) should also be considered. This computation involves an intricate combination of integrals (as we will discuss in Chapter 2, each effect will add 1D or 2D to the integral). Because the integral has no analytic solution, it is usually estimated by numerical methods. Monte Carlo integration, which estimates the pixel values with stochastic point samples

(a) 1209.4 sec. (70 samples/pixel) (b) 1 day and 15 hours (8192 samples/pixel)

Figure 1.3: *Sanmiguel* scene rendered with Monte Carlo path tracing. The rendered image tends to be very noisy when the number of samples is not enough (left), while the converged image takes too much time (right). The two images are rendered on a machine with Intel dual quad-core Xeon E5420 CPU at 2.5 GHz and 32GB of RAM.

in the integral domain, is a popular solution for solving this problem. It is robust for estimating high-dimensional integral and is able to handle various types of light transport paths. Unfortunately, Monte Carlo integration is also known to have a relatively slow converging rate ($1/\sqrt{N}$, where N is the number of samples). When the number of samples is not enough, undesirable artifacts (noise, blur, bending) would appear in the rendered images. Figure 1.3 demonstrates an example: even the *Sanmiguel* scene is only moderately complex and neither depth of field nor motion blur are computed, the image is still very noisy after rendering for twenty minutes with standard Monte Carlo path tracing (please refer to Section 2.5.1).

Although in these years a large number of methods have been proposed to improve the efficiency of Monte Carlo methods, the task of generating noise-free images is still very time-consuming. The major reason is the increasing complexity of scene data. To satisfy the demand for higher-quality images, industry-level scenes usually contain millions to billions triangles to model the physical world in detail. Complex illumination models and materials are also employed. Even with more advanced rendering algorithm and graphics hardware, the growth in data size complexes the simulation of light transport and make the

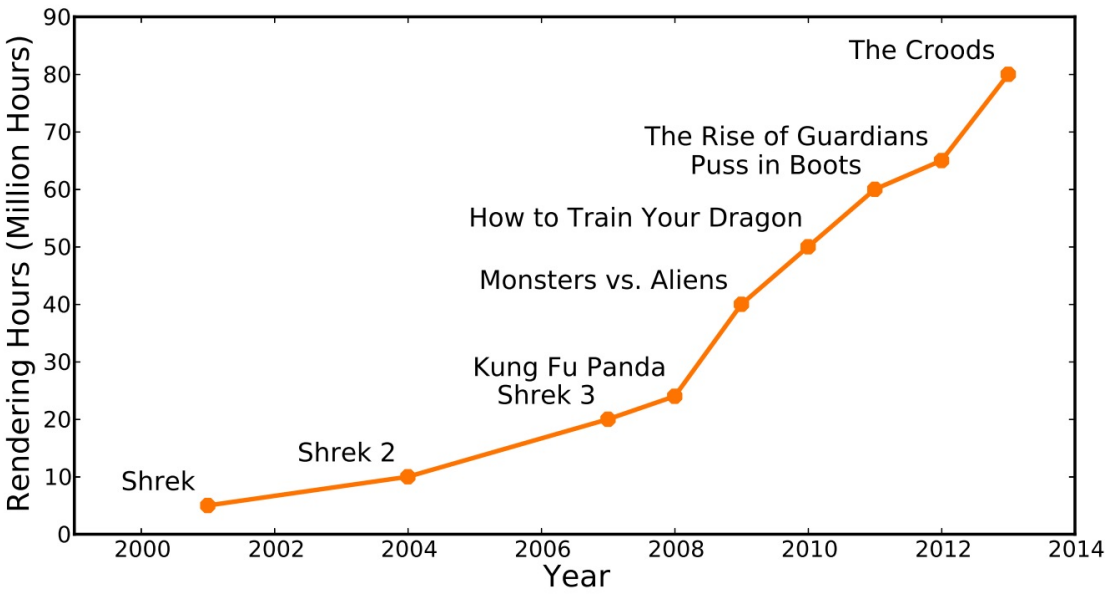


Figure 1.4: The total rendering hours required for DreamWorks' feature movies keep climbing. Image from [79].

rendering process longer. For example, as shown in Figure 1.4, the total rendering hours for DreamWorks Animation's feature movies keep climbing. Therefore, developing more efficient and general rendering algorithms is an important mission that should never be stopped.

In this dissertation we adopt three sampling and reconstruction strategies to accelerate Monte Carlo rendering. First, given a surface point which needs shading, we proposed an algorithm to locate important incoming directions in its hemisphere. By concentrating computational resource on these important directions, we can greatly reduce Monte Carlo noise under a fixed time budget. Second, when rendering translucent materials, we designed a method to locate important surface samples for reducing unnecessary surface irradiance computation. Finally, a general rendering framework based on adaptive sampling and reconstruction was proposed for rendering distributed effects including depth of field and motion blur. We outline the major contributions of this dissertation as follows:

A practical and efficient algorithm for visibility approximation. Shading a sur-

face point requires to evaluate the triple product of incident lighting, material properties, and the visibility (whether a light and a surface point is mutually visible or not). The recent research [11, 45] has shown that visibility is usually the major source of variance (noise) when sampling the shading function. However, due to its expensive evaluation cost, incorporating visibility into variance reduction techniques such as importance sampling is usually impractical. Neglecting visibility dogmatically would make importance sampling less effective, since a large number of occluded lights will be chosen for shading. Our first work, VisibilityCluster, not only can represent visibility more compactly, but also can estimate visibility more efficient. Therefore, it allows visibility to be feasibly incorporated into the importance sampling framework. In Chapter 3, we demonstrate that the proposed method achieves superior noise reduction compared to previous importance sampling techniques.

A new scalable method for rendering translucent materials. Different from surface reflection, translucent materials allow light to enter and scatter within the medium. Although simulating such subsurface scattering gives object a distinct soft look and greatly increases visual richness, it also increases computation tremendously because orders-of-magnitude light paths should be taken into consideration. To accelerate rendering of highly-scattering materials, Jensen *et al.*proposed an analytical dipole diffusion model[58]. It is often rendered with the two-pass approach [57] for reusing computed surface irradiance. Although the two-pass method provides a robust solution, its brute-force irradiance evaluation wastes significantly computational resource in some cases such as close-up rendering. To address this issue, we proposed a dual-matrix representation and a sampling approach. By exploiting the structures of the matrix, we can locate important surface samples and concentrate computational resources on them. In Chapter 4, we show how significant performance improvement is achieved by using our algorithm.

A general framework for adaptive rendering the distributed effects. The process

of rendering involves solving a multidimensional integral. For example, to integrate over a pixel for antialiasing, over the aperture of lens for depth of field, and over the time the camera shutter is open for motion blur. Although Monte Carlo integration provides a general and unified solution for this high-dimensional integral, its slow convergence often demands a large number of samples for producing a visually pleasing image. Our third work is an adaptive rendering framework for reducing the number of required samples in Monte Carlo methods. Performance is improved in two ways. First, given a fixed budget of samples, we determine the optimal sample distribution by concentrating more samples on difficult regions (adaptive sampling). Second, for each pixel, we determine the optimal filter kernel to smooth out noise while preserving image details (better reconstruction). During this adaptive rendering, estimating pixel error is the most difficult task. Our core idea is to apply the Stein's Unbiased Risk Estimator (SURE), a general unbiased estimator for mean squared error (MSE) in statistics, for estimating error. Employing SURE allows our method to use more effective reconstruction kernel rather than isotropic ones (such as the isotropic Gaussians used in previous work [16, 87]). The experiments in Chapter 5 show that our method can better adapt to detailed geometry or high-frequency textures.

The remainder of this dissertation is organized as follows. In Chapter 2 we review the background of light transport, Monte Carlo integration, and some of the most important rendering algorithms. These theorems laid the foundation of our work. The detailed algorithm and implementation details of VisibilityCluster, dual-matrix sampling, and SURE-based adaptive rendering are described in Chapter 3, Chapter 4, and Chapter 5, respectively. Finally, we conclude the dissertation in Chapter 6 and discuss possible future research directions.

Chapter 2

Light Transport, Monte Carlo, and Path Integration

In this chapter, we first review the mathematical formulation of light transport (Section 2.1) and Monte Carlo integration (Section 2.2). Next, in Section 2.3 we introduce a technique, importance sampling, which is commonly used to reduce the variance of a Monte Carlo estimator. In Section 2.4, we translate the rendering problem into a path integral formulation. We also discuss how to apply Monte Carlo integration for solving the integral. Finally, in Section 2.5, we introduce two path sampling algorithms, path tracing and virtual point lights, which laid the foundation of the work in this dissertation.

2.1 Light Transport

Given the description of a 3D virtual world, physically-based rendering simulates the light transport based on physics and mathematics. The rendering equation [59] is one of the most important mathematical formulation for describing the behavior of light-surface interaction. Equation 2.1 shows its hemispherical formulation:

$$L(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega L(x,\omega_i) f_r(x,\omega_o \leftarrow \omega_i)(N(x)\cdot\omega_i)d\omega_i, \tag{2.1}$$

where $L(x, \omega_o)$ is the outgoing radiance at surface point x along direction ω_o. $L_e(x, \omega_o)$ is the emitted radiance (non-zero if x is on a light source) at surface point x along direction ω_o. $f_r(x, \omega_o \leftarrow \omega_i)$ is the bidirectional reflectance distribution function (BRDF) which describes the material property at location x on an opaque surface. Its value represents the ratio of reflected radiance exiting along ω_o to the irradiance incident on the surface from direction ω_i. Finally, the $N(x)$ is the surface normal at location x. Please note that in Equation 2.1 we assume that lights enter and leave at the same surface point x. Only reflection is considered here (opaque surface). For translucent materials, BRDF cannot model subsurface scattering correctly and should be replaced with a more general bidirectional subsurface scattering distribution function (BSSRDF). We will discuss BSSRDF in chapter 4.

Because the L in Equation 2.1 appears in both sides of the equals, the equation is recursive when multi-bounce illumination is taken into account. Recursively expanding the equation results in a multi-dimensional equation. Because there is no analytical solution for the rendering equation, numerical methods are usually adopted to solve the integral. The complexity for solving it depends on the complexity of lights, materials (BRDF), geometry, and the implicit visibility relationship between scene objects. In next section, we will introduce the Monte Carlo integration, which is the only effective solution for solving this high-dimensional integral.

2.2 Monte Carlo Integration

Monte Carlo integration can robustly estimate high-dimensional integral. Suppose we would like to evaluate the following integral:

$$I = \int_{\Omega} f(x) dx. \tag{2.2}$$

MC integration approximates the integral with a set of stochastic samples and combines their results with an estimator. Specifically, suppose the N samples X_1, X_2, X_3, $\cdots$, X_N are independently generated from a probability density function (pdf), $p(x)$, Monte Carlo integration approximates the Equation 2.2 by:

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \tag{2.3}$$

It is not difficult to show that the expected value of this estimator is exactly equal to I:

$$\begin{aligned} E[\hat{I}_N] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}N\int_{\Omega}\frac{f(X_i)}{p(X_i)}p(X_i)dx \\ &= \int_{\Omega}f(x)dx \\ &= I \end{aligned} \tag{2.4}$$

It is worth noting that Equation 2.4 shows that the Monte Carlo estimator is unbiased, which means it will converge to the correct solution as the number of samples is enough. If the expected value $E[\hat{I}_N]$ is not the same as the correct solution, we call the estimator is biased and its bias is computed as $B[\hat{I}_N] = E[\hat{I}_N] - I$. In Section 2.5, we will introduce one biased approach, virtual point light. Although biased approaches do not guarantee the solution will converge to the correct one, they are usually able to produce a visually pleasing image in less time.

The variance of the Monte Carlo estimator is computed as:

$$\begin{aligned} V[\hat{I}_N] &= V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N^2}V\left[\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N}V\left[\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}V\left[\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}\int_{\Omega}\left(\frac{f(X_i)}{p(X_i)}-I\right)^2 p(X_i)dx \end{aligned} \tag{2.5}$$

As the number of samples N increases, the variance of the estimator decreases linearly with N, and the error decreases linearly with $\sqrt{N}$. The root causes the relatively slow converging rate of Monte Carlo estimators. Plenty of variance reduction techniques, including stratified sampling, importance sampling, and control variate, have been proposed to reduce the variance of the estimator and reduce the number of samples needed for convergence. In next section, we will focus on importance sampling because our first work, VisibilityCluster, is built upon it.

Please note the Monte Carlo estimator mentioned above is its one-dimensional formulation. In the one-dimensional case, other integration methods such as midpoint rule, trapezoid rule, and Simpson's rule could converge faster than Monte Carlo integration. However, when the dimensions increases, these methods suffer from the curse of dimensionality (convergence rate grows exponentially with respect to the dimensions). Discontinuity in the rendering function also prevents the use of these methods. On the contrary, Monte Carlo integration can be extended to multi-dimensional cases in a straightforward manner:

$$I = \int\int f(x,y)dxdy \rightarrow \hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i,Y_i)}{p(X_i,Y_i)} \tag{2.6}$$

The unbiased characteristic (Equation 2.4) and derivation of variance (Equation 2.5) still hold in this multi-dimensional case. The variance (and error) of the estimator is

independent of the dimensionality of the integral. For these reasons, Monte Carlo is the only feasible approach for solving the rendering problem.

2.3 Importance Sampling

Importance sampling is an effective strategy to reduce the variance of Monte Carlo integration. Its main idea is that the variance of the Monte Carlo estimator (Equation 2.3) reduces quickly if the samples are drawn from a non-uniform distribution $p(x)$ that is similar in shape to the integrand $f(x)$. In an ideal case, if $p(x)$ is proportional to $f(x)$, we can have an estimator without variance. To derive this fact, suppose we have $p'(x) = cf(x)$, where c is a constant, we first have

$$\int_\Omega p'(x)dx = \int_\Omega cf(x) = 1, \tag{2.7}$$

and we can easily obtain

$$c = \frac{1}{\int_\Omega f(x)dx} = \frac{1}{I} \tag{2.8}$$

When we use $p'(x)$ to generate samples, every estimate in Equation 2.3 will have the same (correct) value:

$$\frac{f(X_i)}{p'(X_i)} = \frac{f(X_i)}{cf(X_i)} = \frac{1}{c} = I, \tag{2.9}$$

meaning that we can perfectly estimate I using only one sample. However, in practice we cannot have such $p'(x)$ because we do not know the value of $f(x)$ without evaluating it. Fortunately, if $p(x)$ is similar to $f(x)$, the variance of the estimator still can be significantly reduced. If the distribution used for sampling is poorly match to the integrand, however, variance will increase. Table 2.1 shows such an example: to sample a function $I = \int_0^4 xdx = 8$ and reach an error of 0.008, the number of samples climbs if the samples are drawn from a poor distribution.

Importance sampling techniques are proposed to choose $p(x)$ that is as similar to

sampling function	variance	samples need for standard error of 0.008
(6-x)/16	56.8/N	887,500
1/4	21.3/N	332,812
(x+2)/16	6.4/N	98,432
x/8	0	1

Table 2.1: Sampling with a distribution which poorly matches to the integrand will increase variance. N is the number of samples.

$f(x)$ as possible. In the case of rendering, the integrand is the triple-product of incident lighting, BRDFs, and the visibility function between the light and surface. Therefore, we prefer to generate samples based on the triple-product. Incoming directions with higher contributions should be sampled more frequently. Efficient construction of a distribution which is similar to the triple product is challenging. It requires efficient and accurate approximation of lighting, BRDFs, and visibility. When not all of these approximations are available, some of them are used to construct $p(x)$. In Section 3.2, we will provide a detailed review of previous importance sampling techniques.

2.4 Path Integration

2.4.1 Path integral formulation

Ray-tracing approaches render images by tracing rays and simulating their paths in scenes. The color of a pixel can be computed by accumulating contributions of all possible paths which start from the light sources, interact (bounce) with scene surfaces, and finally locate on a pixel on the image plane. For this reason, the hemispherical formulation of the rendering equation introduced in Equation 2.1 is usually transformed into an alternative surface formulation. In this section we will introduce this reformulation. By representing the rendering problem as a path integration, we can employ the Monte Carlo integration introduced previously as a numerical solution for solving the integral.

We first define the radiance transported from one surface point x' to another one x as

$$L(x \leftarrow x') = L(x, \omega_o) \tag{2.10}$$

where ω_o is the unit vector from surface point x' to x. It is defined as $\frac{x-x'}{\|x-x'\|}$. Similarly, the surface reflectance function (BRDF) at surface point x' can also be represented as

$$f_r(x \leftarrow x' \leftarrow x'') = f_r(x, \omega_o \leftarrow \omega_i) \tag{2.11}$$

where ω_i is the unit vector from surface point x'' to x'.

When transforming the hemispherical form in Equation 2.1 to surface form, the solid angle is converted to a geometric term which consists of the projection product and a Jacobian:

$$G(x' \leftrightarrow x'') = V(x' \leftrightarrow x'') \frac{|(N(x') \cdot \omega_i)(N(x'') \cdot -\omega_i)|}{\|x' - x''\|^2} \tag{2.12}$$

where $N(x)$ is the surface normal at x. $V(x' \leftrightarrow x'')$ is the binary visibility function to determine whether two points x' and x'' are mutually visible ($V = 1$ means visible, 0 otherwise). By substituting the above terms into Equation 2.1, we obtain the surface form of rendering equation. The outgoing radiance is now computed by integrating over surfaces in the scene:

$$L(x \leftarrow x') = L_e(x \leftarrow x') + \int_A L(x' \leftarrow x'') f_r(x \leftarrow x' \leftarrow x'') G(x' \leftrightarrow x'') dA(x'') \tag{2.13}$$

where A is the union of all surface area in the scene and $dA(x'')$ is the area measure of x''.

By recursively expanding Equation 2.13, we can obtain the following equation:

$$\begin{aligned} L(x_0 \leftarrow x_1) &= L_e(x_0 \leftarrow x_1) \\ &+ \int_A L_e(x_1 \leftarrow x_2) f_r(x_0 \leftarrow x_1 \leftarrow x_2) G(x_1 \leftrightarrow x_2) dA(x_2) \\ &+ \int_A \int_A L_e(x_2 \leftarrow x_3) f_r(x_1 \leftarrow x_2 \leftarrow x_3) G(x_2 \leftrightarrow x_3) \\ &\times f_r(x_0 \leftarrow x_1 \leftarrow x_2) G(x_1 \leftrightarrow x_2) dA(x_2) dA(x_3) \\ &+ \cdots \\ &= \sum_{k=1}^{\infty} \int_{A^{k-1}} L_e(x_{k-1} \leftarrow x_k) \\ &\times \left(\prod_{i=1}^{k-1} f_r(x_{i-1} \leftarrow x_i \leftarrow x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \right) dA(x_2) \cdots dA(x_k). \end{aligned} \tag{2.14}$$

In Equation 2.14, each $x_i \in A$ is a vertex in a path. One path with length k is defined as:

$$\bar{x} = x_0 x_1 \cdots x_k \tag{2.15}$$

where $1 \leqslant k < \infty$. In a complete path, x_k is a point on the light source and x_0 is a the camera.

To apply the Monte Carlo estimator introduced in Section 2.2 for solving this integral, each measurement in Equation 2.14 should be represented as the form in Equation 2.2. We define the measure of a path as a product of measures of surface area of each vertex

$$d\mu_k(\bar{x}) = dA(x_2) \cdots dA(x_k) \tag{2.16}$$

Please note that we do not have $dA(x_0)$ and $dA(x_1)$ in this measure because these two vertices are not created by path sampling. We rewrite Equation 2.14 as:

$$L(x_0 \leftarrow x_1) = \sum_{k=1}^{\infty} \int_{A^{k-1}} f_k(\bar{x}) d\mu_k(\bar{x}) \tag{2.17}$$

where

$$f_k(\bar{x}) = L_e(x_{k-1} \leftarrow x_k) \times \left(\prod_{i=1}^{k-1} f_r(x_{i-1} \leftarrow x_i \leftarrow x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \right) \tag{2.18}$$

Define I^k for the measurement of paths of length k:

$$I^k = \int_{A^{k-1}} f_k(\bar{x}) d\mu_k(\bar{x}) \tag{2.19}$$

and I^* for the measurement of all paths:

$$I_* = \sum_{k=1}^{\infty} I^k \tag{2.20}$$

The integral of different path length k now are independent to each other. As a result, we are able to approximate I^k as

$$\bar{I}_N^k = \frac{1}{N} \sum_{i=1}^{N} \frac{f_k(\bar{x})}{p_k(\bar{x})} \tag{2.21}$$

where $p_k(\bar{x})$ is the probability density function of generating a specific path $\bar{x}$.

To compute the integral using above Monte Carlo estimator, the values of $f_k(\bar{x})$ and $p_k(\bar{x})$ are required. The contribution of $f_k(\bar{x})$ can be evaluated using the rendering equation. The computation for $p_k(\bar{x})$ is more complex because it is also depends on how the path is generated. In ray-tracing approaches, we usually generate a path by sampling a sequence of surface locations as vertices or connecting two existing paths. In next section, we will introduce two popular approaches for path generation.

2.4.2 Antialiasing, depth of field, and motion blur

The number of paths arrived at a specific pixel increases when rendering involves simulation of depth of field or motion blur. In these cases, we have to integrate over the aperture of camera lens and the time camera shutter is open. Another similar case which increases the number of paths is to integrate over the pixel area for dealing with the aliasing issues. Considering all these additional paths, the integral we have to compute for a pixel (i, j) on the image plane becomes

$$I(i,j) = \int_{i-1/2}^{i+1/2} \int_{j-1/2}^{j+1/2} \int_{-1}^{1} \int_{-1}^{1} \int_{t_0}^{t_1} f(x,y,u,v,t) dt dv du \cdots dy dx \tag{2.22}$$

where u and v are the random samples on the aperture of the camera lens and t is the random sample for time. $f(x,y,u,v,t)$ represents the specified rendering configuration for a given set of parameters u, v, t. When the dimension of the integral increases, the number

of samples Monte Carlo methods need for convergence grows significantly. In Section 5.2 we will review some previous methods for efficiently rendering these effects. Our third work, SURE-based adaptive rendering, is also developed for addressing this issue.

2.5 Path-based Rendering Algorithms

Different rendering algorithms use different strategies to generate the sampled paths. In this section, we introduce two popular algorithms which aid solving the path-sampling problem: path tracing and virtual point light methods. The two methods laid the theoretical foundation of our new rendering algorithms in this dissertation.

2.5.1 Path tracing

Path tracing was proposed by Kajiya [59] in 1986. Because of its simplicity and robustness, it has received much acclaim in recent industry applications and feature movie production. The classical path tracing algorithm proceeds as follows: Starting from camera, a path is generated by tracing a ray through a pixel on the image plane. The ray would bounce multiple times in the scene and finally connect to a location on a light source. Every light-surface interaction generates a vertex in the path. For efficiency, shorter paths are usually reused to form longer paths. Path tracing is an unbiased algorithm, however; it tends to generate high-frequency noise when the number of samples is not enough.

Path tracing constructs paths from the camera because only paths passing through image plane are important. Simulating other paths that will not locate on the pixels wastes computational resource. However, in some cases, generating paths starting from the light sources would reduce variance (e.g.caustics). This is called light tracing. Veach combined the two strategies and proposed bidirectional path tracing [105]. The algorithm generates two subpaths each time: one starts from the light and another from the camera. The two subpaths are connected to form complete paths. In general, bidirectional path tracing is

more robust and efficient than classical path tracing and light tracing because it inherits the advantages of both algorithms.

2.5.2 Virtual point light

The seminal work, Instant Radiosity, proposed by Keller [60] laid the foundation of using a set of virtual point lights (VPLs) to represent complex illumination. It is a two-pass algorithm. In the first pass, a set of photons S are traced from the light sources and bounce in the scene. Every time a photon hits a non-specular surface, a VPL is deposited at the location associated with its path contribution so far. The position, normal, and BRDF at this surface location are also recorded. Note that because the photons might have bounced several times, it therefore can be treated as a rough approximation of global illumination. In the second pass, length-one subpaths are generated from the camera (associated with shading points). These paths are connected to the virtual point lights to create complete paths. For each shading point x, the outgoing radiance is estimated with these complete paths. It can be seemed as gathering contributions of all VPLs:

$$L(x, \omega_o) = \sum_{x_i \in S} L_i(x_i) f_r(x \leftarrow x_i, \omega_o) V(x_i) G(x_i) \cos(\theta_i), \tag{2.23}$$

where V is the binary visibility function between the VPL x_i and surface point x ($V = 1$ if visible). $G(x_i)$ is the subtended solid angle at x_o by x_i. θ_i is the angle between the surface normal and the direction from x_i to x. Note in this equation we decouple the visibility term from the geometric term in Equation 2.12.

One of the advantages of using VPLs for rendering is that it provides a unified approach to handle direct and indirect illumination. In addition, performance is improved by reusing subpaths which starts from the light sources (VPLs). The highly correlated paths also makes the result images smoother, reducing the high-frequency noise generated by path tracing. However, virtual point light methods also have their disadvantages. When the number of VPLs is not enough, bending artifacts of lighting features (e.g.shadow

boundaries) appear in the rendered image. Moreover, they are known to suffer from singularities, which results in bright spikes in the image. Singularities occur when the BRDF or geometric term have extremely large contributions. It is usually observed at the corners of walls or on the glossy surfaces. Several approaches have been proposed to reduce the undesired singularities [63, 53, 29, 78].

High-quality rendering requires a large number of VPLs to represent the complex illumination. Accumulating all VPLs for each shading point is often impractical. Lightcuts was proposed to reduce the number of shading operations by adaptive clustering [107, 106, 108]. For each point, a shading cut is used to keep the expected error under a threshold. Visibility is not included when estimating the error bound. Hasan *et al.* [54] interpreted the many-light problem using a matrix form. They exploited the low-rank property of the matrix for efficient rendering by sampling both rows (shading points) and columns (lights). Ou and Pellacini [80] further observed that matrices for local surfaces tend to have very low ranks. They computed per-slice light clusters for more efficient approximation.

There are also VPL methods focusing on real-time applications [26, 27, 76]. They often ignore occlusion when computing indirect lighting. To take account of visibility, Ritschel *et al.* [84] proposed the Imperfect Shadow Map to accelerate the construction of shadow maps for VPLs. Dong *et al.* [34] clustered VPLs into large area lights and determined visibility using soft shadowing techniques. Bashford-Rogers *et al.* [7] used visibility grids to approximate geometry and accelerate multi-bounce indirect illumination. For efficiency, these methods pursue visually pleasing results rather than physically accurate ones. They are validated by a perceptual study [113] in the context of interactive VPL-based rendering.

Chapter 3

VisibilityCluster: Average Directional Visibility for Many-Light Rendering

As discussed in Section 2.3, importance sampling is a commonly employed technique for reducing Monte Carlo noise. Its main idea is to sample with a distribution proportional to the integral kernel, the triple product of incident lighting, BRDFs, and visibility. To construct such distributions, we require accurate approximation of these terms. In this chapter, we focus on visibility and introduce our VisibilityCluster algorithm for efficient visibility approximation and representation. We demonstrate that the proposed method allows visibility to be incorporated into importance sampling at a reasonable cost, significantly reducing variance in Monte Carlo rendering. In addition, VisibilityCluster can also be used to increase realism of local shading by adding directional occlusion effects. Experiments demonstrate that our method significantly outperforms the state-of-the-art importance sampling algorithms and successfully enhances the preview quality for lighting design.

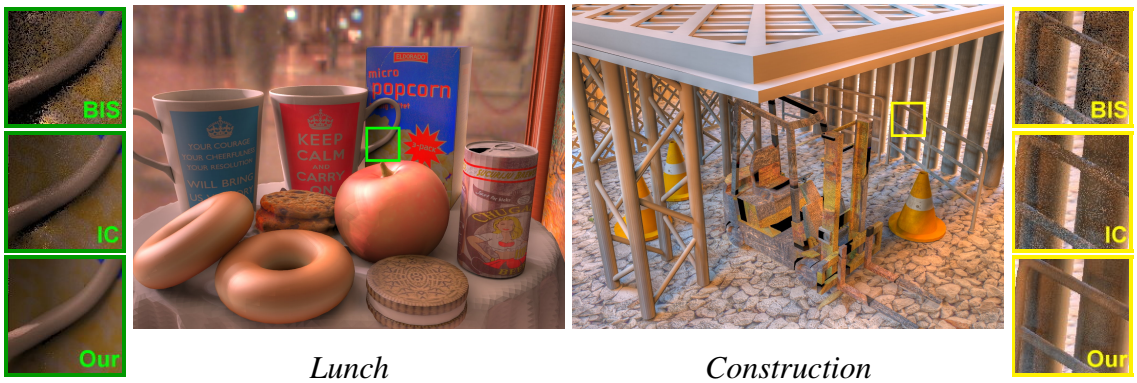


Figure 3.1: Examples for limitations of previous importance sampling methods. Previous methods produce significant noise in regions with high-frequency visibility variations due to complex occlusion or fine scene structures, such as the coffee mug handle very close to the popcorn box in the *Lunch* scene and the thin handrails in the *Construction* scene. Insets show detailed comparisons for parts of the scenes rendered with different methods given the same amount of time (200 seconds for *Lunch* and 500 seconds for *Construction*). The insets, from top to bottom, are Bidirectional Importance Sampling (BIS) [109], Importance Caching (IC) [45], and our approach. Our method provides better noise reduction across the whole image including these difficult areas.

3.1 Overview

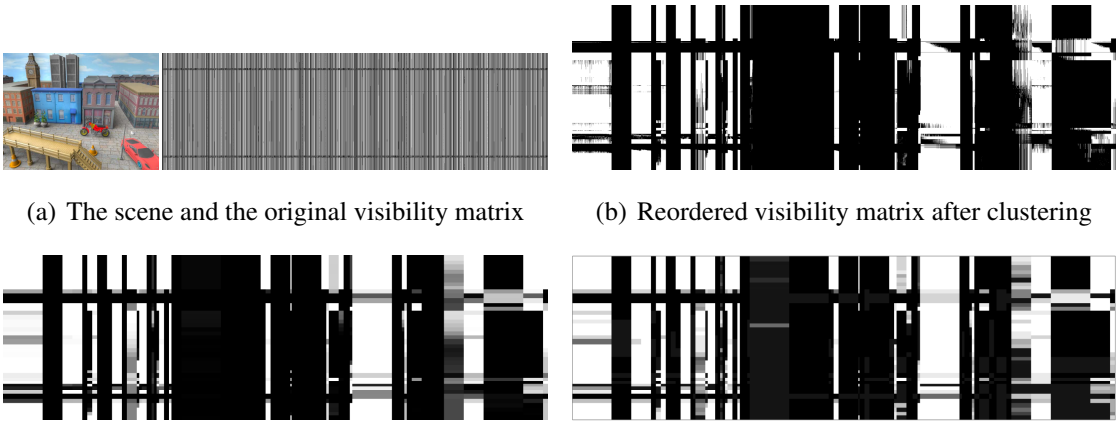
The commonly employed Monte Carlo ray tracing method tends to suffer from noisy results due to the large variance in stochastic sampling. Importance sampling is an effective strategy for reducing that variance. It requires to efficiently approximate the integral kernel of the rendering equation, a triple product of incident lighting, material properties and visibility. Among these terms, visibility is often ignored because its estimation demands expensive shadow ray casting or shadow map construction. However, a recent study has reported that in industry-level scenes more than half of the shadow rays (sometimes over ninety percent of them) end up with occlusion [11]. Thus, the neglect of visibility largely limits the effectiveness of importance sampling.

We propose an efficient method for estimating average visibility to improve the effectiveness of importance sampling. We focus on rendering with the many-light formulation

(i.e., the virtual point light method discussed in Section 2.5.2), which has received much attention in recent years. In previous importance sampling approaches, visibility is either completely omitted [109] or sparsely sampled [45] because of its expensive computational cost. The bidirectional importance sampling (BIS) approach [109] only approximates lighting and BRDFs without taking visibility into account. It suffers from significant noise when contributions from strong lights or BRDF peaks are occluded. Georgiev *et al.* [45] later proposed Importance Caching (IC) to address this problem. However, their method fails to handle high-frequency variations of geometry and visibility. Figure 3.1 shows problems with these approaches.

Our method, VisibilityCluster, provides a method for efficient computation and compact representation of the visibility function. The method is based on the observation that visibility terms in the many-light transport matrix exhibit good local structures if shading points and lights are properly clustered. Figure 3.2(a) visualizes the visibility matrix for a scene, in which rows and columns correspond to shading points and lights respectively, and matrix entries represent their visibility values. It took 4,856 seconds to compute the full matrix of pairwise visibility. After clustering, the reordered matrix exhibits good local structures (Figure 3.2(b)). We call a submatrix formed by a cluster of lights and a cluster of shading points a VisibilityCluster and calculate its average visibility (Figure 3.2(c)). The average visibility is particularly useful for the applications that do not require perfect visibility, such as importance sampling. By sampling lights and shading points, we can estimate all average visibility terms between clusters in 14 seconds and with only 3.57% error (Figure 3.2(d)). Our method has several advantages:

- It is usually sufficient to estimate the average visibility accurately with few visibility samples.
- It makes the importance function more compact and improves performance for both construction and sampling, making it more scalable in terms of number of lights.



(a) The scene and the original visibility matrix

(b) Reordered visibility matrix after clustering

(c) Average visibility with clustering

(d) Approximated average visibility with our method

Figure 3.2: Visualization of the visibility matrix for the TOWN scene. The rows and columns correspond to shading points and lights, respectively. Each matrix entry represents the visibility between the corresponding shading point and light. In the original order of shading points and lights, the visibility matrix (a) has little coherence. After reordering, the visibility matrix (b) exhibits blocky structures due to visibility coherence. We call a submatrix formed by a light cluster and shading cluster a VisibilityCluster. By averaging the visibility terms within each VisibilityCluster, we obtain the average visibility matrix (c). It took 4,856 seconds to compute pairwise visibility between each light and shading point, and construct the average visibility matrix. The brightness indicates the visibility ratio between a pair of clusters. Our VisibilityCluster algorithm efficiently approximates the average visibility matrix by only casting few shadow rays for each VisibilityCluster. The approximated average visibility matrix (d) with VisibilityCluster was estimated with only 14 seconds and 3.57% error to the average visibility matrix (c).

- The estimated visibility provides a direct quality measure to visibility estimation and can be used for guiding further refinement of clustering.

The proposed VisibilityCluster method produces visibility information and can be built on top of bidirectional importance sampling (BIS) [109] for more effective variance reduction. Including visibility allows us to better sample unoccluded lights. Experiments show that our method achieves superior noise reduction, compared to other state-of-the-art importance sampling techniques when rendering a variety of complex scenes with different illumination conditions. In addition to importance sampling, we demonstrate

that VisibilityCluster can be combined with local shading to produce visually pleasing directional occlusion effects. Altogether, the proposed VisibilityCluster algorithm offers a good compromise between performance and quality, and is well suited for lighting previews and high-quality rendering.

The remainder of this chapter is organized as follows. We first review previous research in visibility sampling and approximation in Section 3.2. Then in Section 3.3 we describe the problem formulation of applying importance sampling for noise reduction in the many-light problem. Next, we give the algorithm and implementation details of our VisibilityCluster algorithm in Section 3.4. In Section 3.5, we describe how to integrate VisibilityCluster into importance sampling framework and directional occlusion. We also conduct detailed comparisons between previous approaches and our method. Finally, Section 3.6 concludes this chapter and introduce some possible future research directions.

3.2 Related Work

In this subsection we review previous research in importance sampling for noise reduction, visibility algorithms for sampling and approximation, and directional occlusion for shading enhancement. For the many-light formulation and its related approaches, please refer to Section 2.5.2.

3.2.1 Importance sampling

In section Section 2.3 we introduced the major concept of importance sampling. It reduces the variance of Monte Carlo estimators by sampling with a distribution proportional to the integral kernel, which in our case is the triple-product of incident lighting, BRDFs, and visibility. However, efficient construction of such distributions is challenging since it requires accurate approximation of these terms. Among them, visibility is particularly costly to approximate.

Earlier importance sampling algorithms only approximate one of these terms, such as illumination [1, 64, 98] or material properties [65]. In most cases both lighting and surface BRDFs are complex and high-frequency, sampling based on illumination or material properties only would not be efficient enough. The multiple importance sampling strategy proposed by Veach [104] draws samples from illumination and BRDF independently. The two kinds of samples are combined to produce better sampling efficiency. Recently, several (bidirectional) product importance sampling methods have been proposed. Pioneered by wavelet importance sampling [22], earlier approaches focus on sampling the product of environment lighting and BRDFs [24, 21, 56]. Wang and Åkerlund [109] later proposed a more general approach that can sample both structured and unstructured illumination (such as area lights or indirect lighting) by using the many-light formulation. Other well-known approaches for generating product distributions of lighting and BRDFs are based on the idea of importance resampling [13, 100]. They approximate the product distribution by starting from one distribution and then refining with the other. These approaches can handle both direct and indirect illumination. The major problem of the above methods lies in the oversampling of occluded lights, making them unsuited for scenes with complex visibility.

Some approaches are built on top of product importance sampling and try to reduce visibility variance. Ghosh *et al.* [46] reduced noise in partially occluded regions by using Metropolis-Hastings mutations. Rousselle *et al.* [86] included a very conservative visibility term by approximating scenes with inner spheres. The approach is not suitable for fine geometry and can not handle partially occluded areas. Clarberg *et al.* [20] used control variate to reduce variance. They reformulated the rendering equation with an interpolated visibility term. Although with some success, without changing the distribution for sampling, their method is less effective. Finally, the table-driven adaptive importance sampling proposed by Cline *et al.* [23] improves sample quality by reusing importance

functions from neighbour pixels in the previous rendering process. Their method suffers from large variance near the boundary, where no shading information is available.

Georgiev *et al.* [45] recently proposed importance caching to exploit coherence in the reflection integrand. Full lighting contributions including visibility are evaluated and cached at sparse locations. Several types of distributions, ranging from aggressive to conservative, are built at these locations and reused by nearby shading points with a multiple importance sampling scheme. The major limitation of their method is that the quality of distributions decays quickly in the presence of high-frequency variations of materials, geometry and visibility. In addition, as the number of lights grows, the memory for storing distributions and the computation overhead for generating distributions become big concerns.

3.2.2 Visibility algorithms

Several methods have been proposed to reduce the number of shadow tests in the last twenty years. Agrawala *et al.* [2] exploited image-space coherence to efficiently render soft shadows from area lights. Ben-Artzi *et al.* [9] further extended the idea to reduce shadow tests for environment lighting. Shirley *et al.* [93] accelerated rendering by only sampling the visibility of strong lights. These methods suffer from artifacts when most strong lights are occluded. Hart *et al.* [52] reduced the cost of visibility sampling by identifying the occluded region of a light source. They stored occluders in a map and re-projected them onto light sources for culling out the occluded part. This algorithm observes good performance for simple scenes, but does not scale well with geometry and light complexity.

There are also algorithms for approximating visibility using dedicated graphics hardware or point-based representations. Stewart and Karkanis [99] used item buffer and graph relaxation to construct approximated visibility maps. Dutré *et al.* [39] represented

geometry as point clouds and used them for visibility estimation. These methods can only achieve limited success for complex scenes.

The Local Illumination Environment (LIE) [43] intends to reduce both the frequency and cost of shadow computation. A LIE represents a part of the scene (an octree cell in their implementation), combined with lists of lights that are fully visible, fully occluded, and partially visible during rendering. Occluders for partially visible lights are also recorded. Contributions from fully visible lights are gathered without tracing shadow rays. Visibility tests for partially visible lights are accelerated by only testing with the recorded occluders. LIE also does not scale well when the number of lights and triangles grows. It takes a long time to assort the lights, and significant amount of memory is required to store the occluders.

Donikian *et al.* [35] used an adaptive scheme to iteratively accumulate visibility information. They combine uniform, block-level, and pixel-level probability density functions with different weights at different iterations for light sampling. Their method is very robust; however, it has an expensive start-up cost since uniform sampling is used in early stages. Recently Ramamoorthi *et al.* [83] investigated the impact of sampling patterns on the quality of soft shadows. They offer guidelines for the strategy of sampling area lights with different shapes.

3.2.3 Directional occlusion

Our method can be used for enhancing local shading with directional occlusion. In this context, Ritschel *et al.* [85] proposed a screen space method for interactive applications. It produces visually pleasing results but fails to account for occluders outside the view frustum. The practical filtering method proposed by Egan *et al.* [40], on the other hand, is designed for high quality rendering. Our method strikes a balance between performance and quality, and also offers a good solution for lighting previews.

3.3 Background

In this section we outline the many-light rendering problem which the VisibilityCluster algorithm aids solving. Prior work has shown that indirect illumination, environment lighting, and many direct light sources can all be approximated by many virtual point lights [60, 107, 54]. Assume that all sources of illumination are represented by a set of point lights, S. For a shading point x_o, its reflected radiance L_o along the viewing direction ω can be computed by summing contributions from all lights in S:

$$L(x, \omega_o) = \sum_{x_i \in S} L_i(x_i) f_r(x_i \to x, \omega_o) V(x_i) G(x_i) \cos(\theta_i), \tag{3.1}$$

where x_i is a point light in S, L_i is its incident radiance, f_r is the surface BRDF of x, V is the binary visibility function between x and x_i, and $G(x_i)$ is the subtended solid angle at x by x_i. By combining $G(x_i)$ and $\cos(\theta_i)$ into BRDF and hiding x and ω for a fixed shading point, the equation can be simplified as:

$$L = \sum_{x_i \in S} L_i(x_i) f_r(x_i) V(x_i). \tag{3.2}$$

As the number of lights increases, directly summing the contributions of all lights becomes impractical. Monte Carlo method and importance sampling are often employed to efficiently estimate Equation 3.2 with a small number of samples N:

$$L \approx \frac{1}{N} \sum_{s=1}^{N} \frac{L_i(x_s) f_r(x_s) V(x_s)}{p(x_s)}, \tag{3.3}$$

where p is the probability mass function (importance function) used to draw samples. Variance can be effectively reduced by choosing p that approximates the product of lighting, BRDF and visibility. As it is very expensive to even obtain approximation of visibility, most previous work assumes uniform visibility and only approximates the product $L_i f_r$ for p. Unfortunately, the constant visibility assumption often fails for complex scenes with non-negligible visibility variations. The proposed VisibilityCluster method can estimate visibility more efficiently. Therefore, it becomes feasible to incorporate visibility

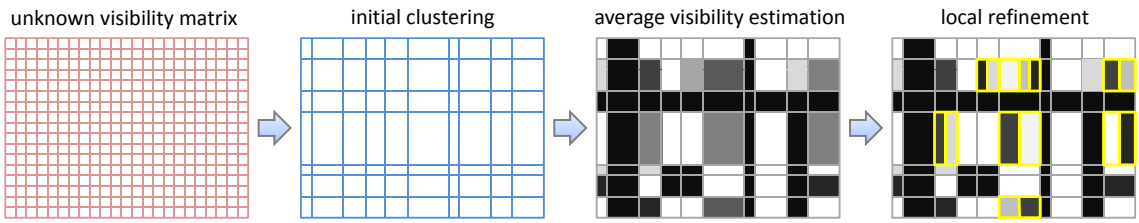


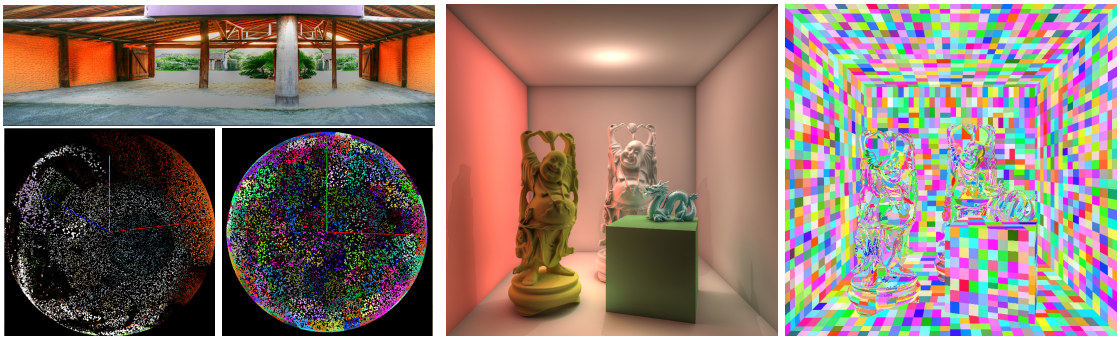
Figure 3.3: Construction of VisibilityClusters. We first form initial VisibilityClusters by clustering the lights and shading points based on their geometric properties independently. Next, we estimate the average visibility of each submatrix (for a pair of light and shading clusters) by sampling few shadow rays. Finally, we locally refine the VisibilityClusters with large visibility variance (highlighted in yellow) by splitting light clusters.

into p for more effective variance reduction. We call it triple-product importance sampling since the product $L_i f_r V$ is approximated.

3.4 VisibilityCluster

Our goal is to construct VisibilityClusters with high visibility coherence: all shading points in a shading cluster have similar visibility functions toward the lights in a light cluster. A straightforward way is to start with two large clusters containing all lights and shading points respectively, estimate their visibility variance, and split them into smaller ones if the variance is too large. However, since each iteration of the variance estimation demands expensive shadow testing, the construction cost is not acceptable. To find a good compromise between accuracy and efficiency, we propose a two-pass approach described in this section.

Figure 3.3 demonstrates the flowchart of the two-pass method. In the initial clustering stage, we cluster lights and shading points separately according to their geometric properties. This is based on the observation that locally close shading points often behave similarly to a group of lights within a small solid angle. For each pair of shading point and light clusters, we estimate the average visibility of their VisibilityCluster by sampling



Light clustering Shading point clustering

Figure 3.4: Light clustering and shading point clustering. Left: The original environment map (top) is represented by 32K VPLs (bottom left) which make up 1,200 clusters (bottom right, lights with the same pseudo color belong to the same cluster). Right: Shading points are grouped into 2,400 clusters.

lights and shading points uniformly within each cluster. Finally, based on the estimated average visibility, local refinement is performed for VisibilityClusters with large visibility variance by splitting the light clusters.

3.4.1 Initial clustering

In the initial clustering stage, we separately cluster lights and shading points based on their geometric properties and use those to create initial VisibilityClusters.

Light clustering

We cluster lights using a modified IlluminationCut approach [18, 109] for its native support of hierarchical refinement. The original IlluminationCut starts by building a binary tree for all lights based on locations and directions. Each leaf corresponds to an individual point light while each internal node represents a cluster of lights. For each light cluster L_k, we record its bounding box, the mean $\langle L_k \rangle$ and the variance $\mathrm{Var}(L_k)$ of illumination for all lights within the cluster.

Any cut to the binary tree defines a clustering configuration. In our case, the criterion

for selecting a cut is to keep the variance of both luminance and visibility low. Starting from the root, we keep selecting a node and splitting it into its two children. Since at this stage we do not have the results of visibility testing, there is no guarantee that visibility variance will be reduced by splitting nodes. We use a heuristic approach that works well in practice to partition the tree. Assume we want to create $N_c + 1$ clusters. N_c splits are required. For the first $\lfloor k \times N_c \rfloor$ ($k \leq 1$) splits, we select nodes according to their luminance values. The cluster (node) with the largest luminance variance $\mathrm{Var}(L_k)$ is replaced with its two children in each iteration. For the rest of splits, we focus on splitting clusters with large spatial extents because they potentially have large visibility variations. We select the cluster with the largest axis extent to split until the number of clusters has been reached. In practice, we found that $k = \frac{2}{3}$ works well for all kinds of illumination conditions.

Shading point clustering.

Shading points are clustered in a top-down partitioning manner with a kd-tree as in previous methods [106, 80]. First, we construct a tree for all shading points using 6D coordinates including both positions and surface normals. Next, similar to light clustering, we compute a shading cut by iteratively replacing the node with the largest axis extent with its two children, until reaching the target number of shading clusters. The final shading cut represents the clustering configuration.

We have experimented with other methods, such as k-means clustering, applied to grouping lights and shading points. With the same budget of the cluster size, different clustering methods produce results of similar rendering quality. We adopt the top-down splitting approaches, IlluminationCut and kd-tree, for better efficiency. Figure 3.4 gives examples of light and shading point clustering.

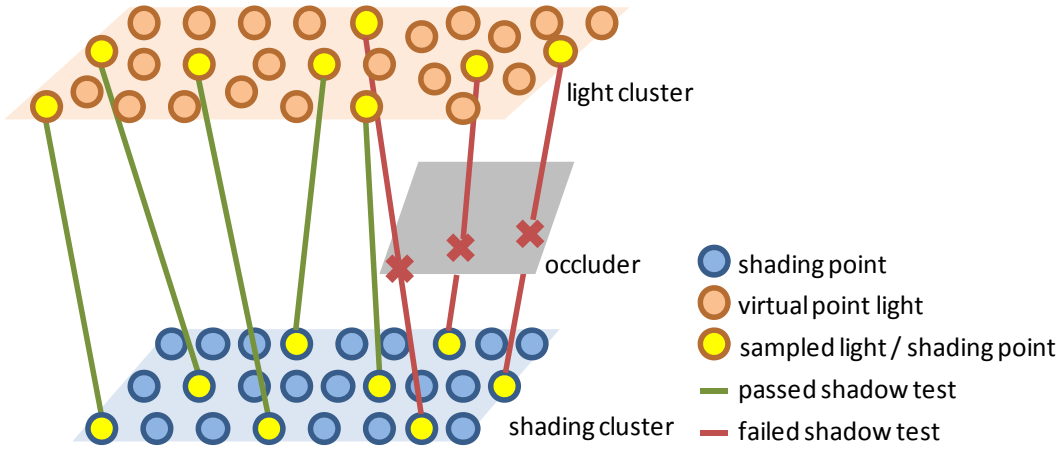


Figure 3.5: Average visibility estimation. For a shading point cluster and a light cluster, we sample shading points and lights uniformly and evaluate their visibility. The average visibility between the two clusters is approximated as the ratio of unoccluded shadows rays to all shadow rays. For this particular example, it is $5/8 = 0.625$.

3.4.2 Average visibility estimation

After clustering, we estimate the average visibility for each VisibilityCluster. Assume there are N lights in the light cluster and M shading points in the shading point cluster, the true average visibility is obtained by performing $N \times M$ shadow tests and counting the ratio of unoccluded rays to all rays which is computationally expensive. Fortunately, because of visibility coherence within a VisibilityCluster, the average visibility can be estimated accurately using only a few shadow rays (in our implementation, usually 10-12). The shadow rays are generated by sampling the lights and shading points uniformly in their clusters (Figure 3.5). The average visibility of a VisibilityCluster is approximated as the ratio of unoccluded shadow rays to all sampled shadow rays.

3.4.3 Local refinement

The estimated average visibility ($\bar{V}$) from the previous step also serves as a quality measure to visibility estimation since it is related to visibility variance. The visibility

variance of a VisibilityCluster C_k is calculated as follows:

$$\mathrm{Var}(C_k) = \frac{1}{n}\sum_{i=1}^{n}(V_i - \bar{V})^2, \tag{3.4}$$

where n is the number of shadow tests and V_i is the individual result of a shadow test. Since $V_i \in \{0,1\}$, we have $V_i^2 = V_i$. In addition, $\frac{1}{n}\sum_{i=1}^{n} V_i = \bar{V}$. As a result, we have

$$\mathrm{Var}(C_k) = \bar{V} - \bar{V}^2 = \bar{V}(1-\bar{V}). \tag{3.5}$$

From the other point of view, if V_i follows a Binomial distribution with the success probability $\bar{V}$, the sample variance is $\bar{V}(1-\bar{V})$. Thus, by using Equation 3.5, we can estimate visibility variance from the estimated average visibility. For VisibilityCluster whose visibility variance is larger than a predefined threshold σ_v ($\mathrm{Var}(C_k) \geq \sigma_v$), the current clustering configuration is not good enough and the visibility estimation might be inaccurate. We replace the light cluster with its two children in the light tree and re-estimate the average visibility for each one. The refinement continues until the visibility estimation for all VisibilityClusters are accurate enough or the maximal depth has been reached. Note that local refinement is performed per shading cluster. Thus, as shown in Figure 3.3, shading clusters (rows in the matrix) could have different light clustering configurations (column configurations). Figure 3.6 demonstrates the improvement of visibility coherence with local refinement.

It is worth mentioning that for local refinement we only split light clusters but not shading clusters because of the cost of building importance functions. For importance sampling, for each shading point, we need to determine its shading cluster and calculate the corresponding cumulative distribution function (CDF) for all light clusters (Figure 3.7(d)). Splitting shading clusters generates additional shading clusters, each requiring its own CDF (Figure 3.7(e)). As a result, the number of shading clusters grows fast with the refinement depth, making the CDF calculations very expensive. Thus, we choose to split the light clusters only (Figure 3.7(c)). In practice, we found that our choice of only

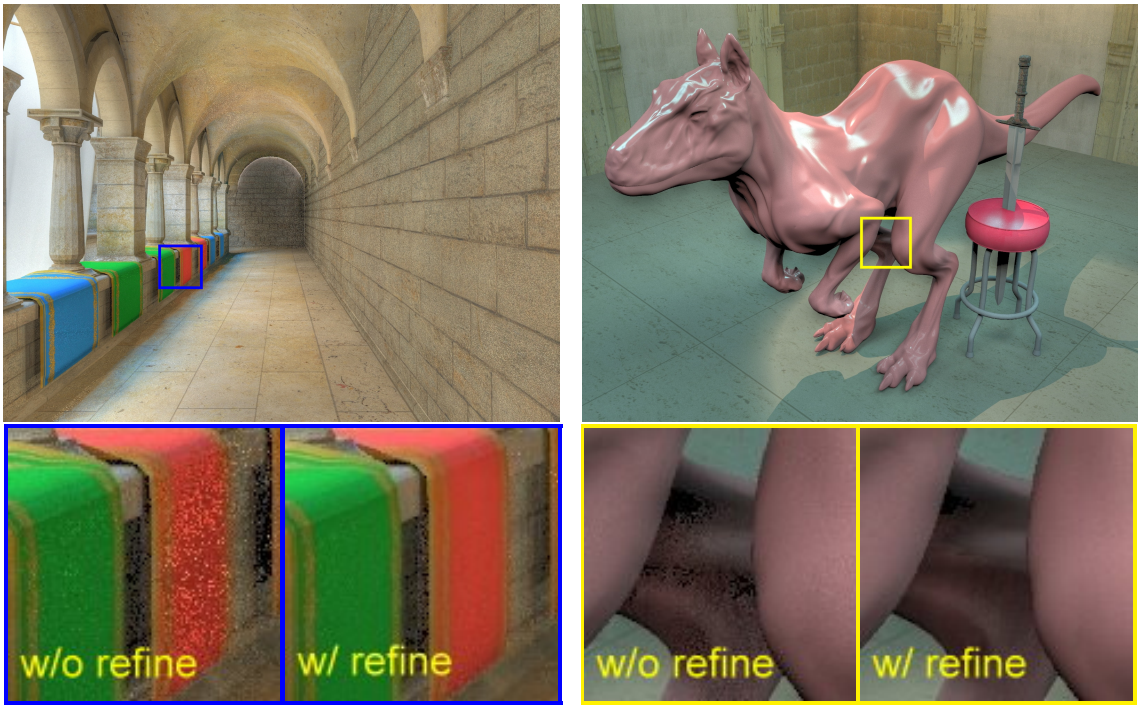


Figure 3.6: Equal-time comparisons of rendering using the initial VisibilityClusters (w/o refine) vs. the locally refined ones (w/ refine). Local refinement improves rendering quality in regions with difficult visibility by reducing visibility variation within a VisibilityCluster.

splitting light clusters has little impact on the quality of VisibilityClusters, but makes CDF construction (Figure 3.7(f)) much more efficient and memory-friendly.

3.4.4 Bias avoidance

For the case that all shadow tests return occluded, assigning zero visibility dogmatically introduces bias. To avoid bias, an additional test is performed to check whether the light cluster falls on the other side of the hemisphere of the shading point cluster. If so, the average visibility is set to 0; otherwise, it is set to a small value ε. We determine ε for a shading cluster by estimating its average accessibility p_v (computed by averaging its visibility ratios to all light clusters). ε is set to $(1-p_v)^n$ where n is the number of performed shadow tests. This value accounts for the probability of failing n times in shadow

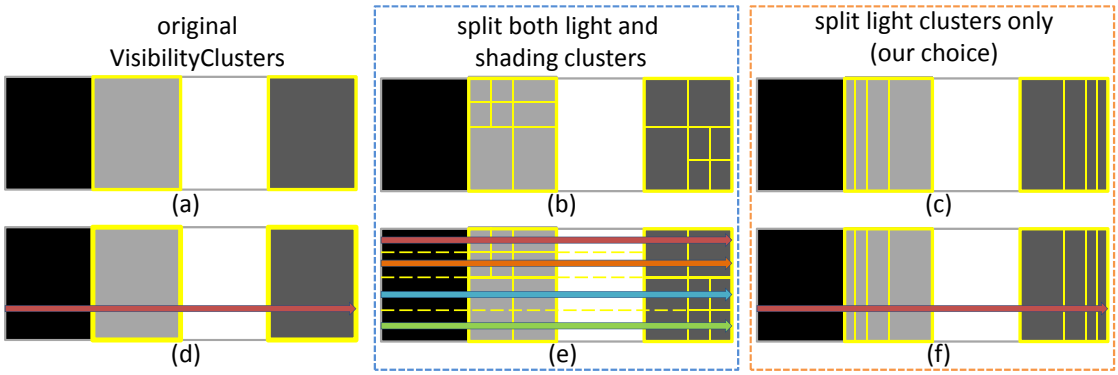


Figure 3.7: The choice of splitting for local refinement. (a) shows the original VisibilityClusters for a shading cluster. We want to refine the second and fourth ones (outlined in yellow). (b) is a refinement of both light and shading clusters. (c) is a refinement of light clusters only. Importance sampling requires calculating a cumulative distribution function (CDF) over all light clusters, as shown in (d). Splitting shading clusters as in (b) would result in additional shading clusters, each with its own CDF (the rows with different colors in (e)). We found splitting light clusters only as in (c) produces results of similar quality but is significantly more efficient and memory-friendly for building importance functions (f).

tests if the average probability is p_v.

3.5 Experiments

We have implemented the proposed VisibilityCluster algorithm on PBRT2 [82] for two applications: triple-product importance sampling and directional occlusion. Experiments were performed on a machine with Intel Xeon 5420 CPU (2.5GHz) and 32 GB RAM.

Table 3.1 lists the scene profiles and rendering settings for all test scenes. The number of initial light clusters was set according to the illumination complexity. For most scenes, it was set to about 1K as suggested by previous work [109]. High-frequency lighting, as observed in the LUNCH scene (Figure 3.10), might require more clusters. Similarly, the number of shading clusters was set based on the geometry complexity. In our ex-

periments, 6,400 clusters usually gave good results. More accurate visibility estimation using more shadow rays allows fewer lights to be sampled for estimating Equation 3.3 in the same amount of time. Thus, we need to find a good tradeoff between the accuracy of the visibility estimate and the number of allowed sampled lights. As demonstrated in Figure 3.8, 8-12 shadow rays provided a good compromise between accuracy and performance. Finally, for local refinement, we set σ_v to 0.16 (equivalently, the refinement is performed if the estimated average visibility is between 0.2 and 0.8) and the maximal refine depth to 2 for all test scenes in this paper.

Scene	Faces	Lights	LC	Avg. LC	Time for VC (Ratio)
Town	818,047	32K	600	697	19.5 sec. (13%)
Lunch	180,807	32K	1,200	1321	24.1 sec. (12%)
Killeroo	744,140	32K	700	748	24.3 sec. (8%)
Sponza	279,163	78K	1,400	1464	108.5 sec. (18%)
Room	462,878	40K	800	914	41.7 sec. (14%)
Construction	272,566	65K	800	935	58.4 sec. (12%)
Hairball	2,893,834	32K	800	972	65.9 sec. (8%)

Table 3.1: Scene profiles and rendering settings. The columns list the number of triangles, the number of virtual point lights, the number of initial light clusters (LC) and the average number of light clusters after local refinement (Avg.LC), and the time for computing VisibilityCluster (and its ratio to the total rendering time).

3.5.1 Triple-product importance sampling

VisibilityCluster can be easily combined with bidirectional importance sampling (BIS) [109] to offer a more effective triple-product importance sampling solution. We use BIS to obtain bidirectional importance for each light cluster. Then, the estimated average visibility is multiplied to form the triple-product importance function for sampling:

$$p(x_s) \sim |L_k| \langle L_k \rangle \langle \rho_k \rangle \langle V_k \rangle \cdot \frac{1}{|L_k|} = \langle L_k \rangle \langle \rho_k \rangle \langle V_k \rangle \tag{3.6}$$

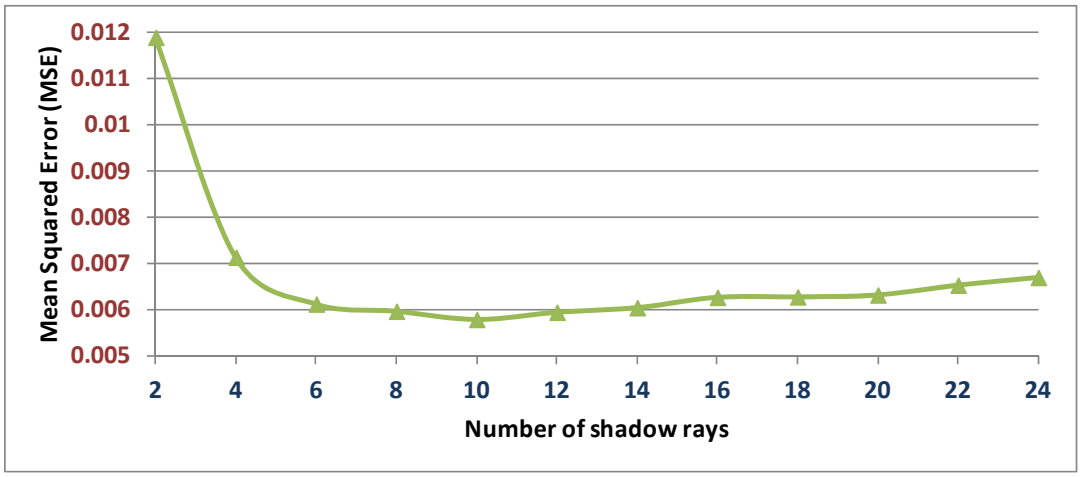


Figure 3.8: The experiment to determine the number of shadow rays for the average visibility estimation in the *Lunch* scene. To find a good balance between computation time and accuracy, we carefully tune shadow ray and sampled light numbers while keeping the rendering time at 200 seconds. The plot of the number of shadow rays vs. MSE indicates that 8-12 shadow rays provide a good compromise.

where $|L_k|$, $\langle L_k \rangle$, $\langle \rho_k \rangle$, $\langle V_k \rangle$ are the number of lights, the average luminance, the average BRDF, and the average visibility of the k-th light cluster respectively. To select a light from the set of VPLs S for shading, we first sample a light cluster using the triple-product importance $|L_k| \langle L_k \rangle \langle \rho_k \rangle \langle V_k \rangle$ and then uniformly sample a light from the selected light cluster.

Comparisons

We compared VisibilityCluster with bidirectional importance sampling (BIS) [109] and importance caching (IC) [45], which are state-of-the-art product and triple-product importance sampling methods respectively. The number of sampled lights for each shading point was carefully set for each method to provide equal-time comparisons, including the time for computing importance records in IC and estimating average visibility in VisibilityClusters. As suggested by the importance caching authors, the number of importance records was set to 7K and the number of nearest records used by multiple

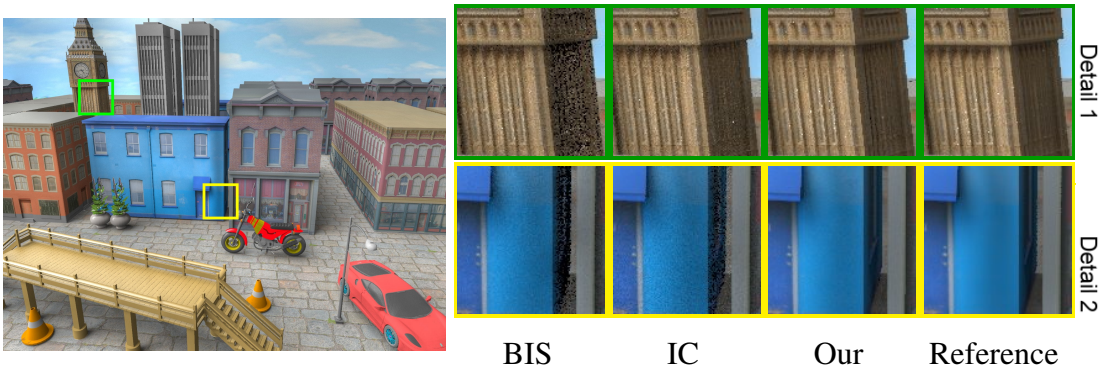


Figure 3.9: *Town* (equal-time comparison, 150 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. Our method provides better noise reduction where there is severe occlusion (detail 1) or high-frequency geometry variation (detail 2).

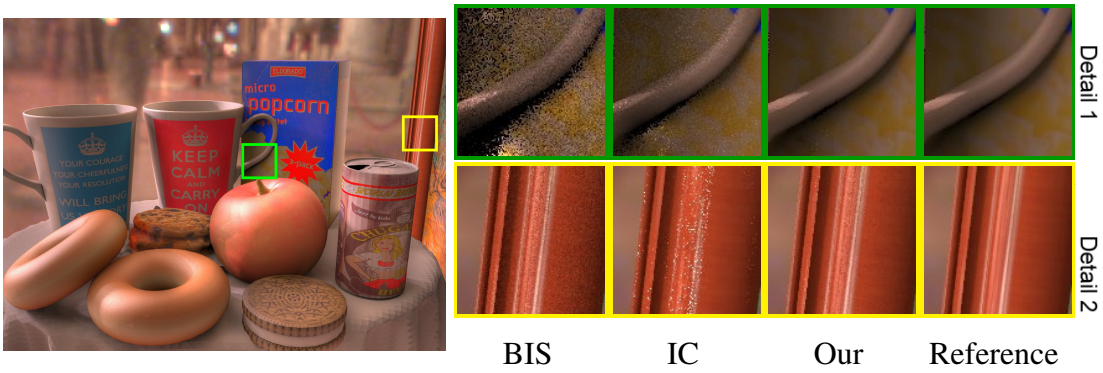


Figure 3.10: *Lunch* (equal-time comparison, 200 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. BIS produces severe noise due to occlusion of strong lights (detail 1). IC cannot capture well high-frequency BRDFs and fine geometry details due to its sparse records (detail 2).

importance sampling was set to 3.

The first two scenes, *Town* (Figure 3.9) and *Lunch* (Figure 3.10), were rendered using direct lighting from environment illumination. *Town* was illuminated by a low-frequency cloudy sky and *Lunch* was illuminated by high-frequency indoor lighting. For *Town*, BIS suffers from severe variance on the clock tower (Figure 3.9, detail 1). Tall buildings act as occluders to each other and block the sun in the sky. BIS wastes most light samples toward

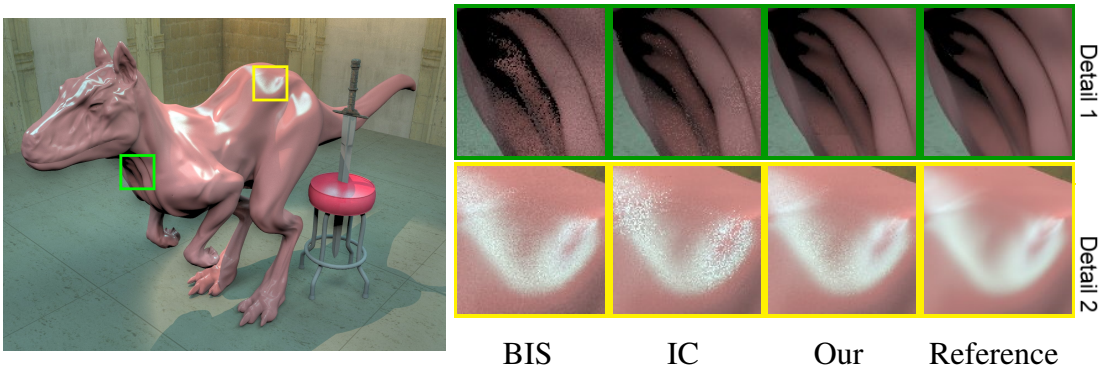


Figure 3.11: *Killeroo* (equal-time comparison, 300 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. BIS fails in regions with complex visibility variation (detail 1). IC is inefficient for glossy materials (detail 2). Our method is more effective for both cases.

the occluded sun since it has the strongest luminance. IC reduces variances by including the visibility term into the importance function. However, in the narrow passageway with only few importance records (Figure 3.9, detail 2), the rendering looks quite noisy because the nearby records provide inaccurate information. Similar situations occurred in *Lunch*. The image rendered with BIS is very noisy since bright lights are occluded by the oil painting (Figure 3.10, detail 1). IC outperforms BIS in most areas, but still observes significant noise in areas with large visibility variation, e.g. the partially occluded regions on the table (Figure 3.16). Its sparse records also cannot capture the fine geometry variation on the painting's frame (Figure 3.10, detail 2). Our approach alleviates these problems by properly clustering lights and shading points to better exploit the coherence in visibility.

The third scene, *Killeroo* (Figure 3.11), was illuminated by a bright spot light and a skylight environment map. The scene demonstrates a very difficult case because most illumination is occluded by the surrounding walls. It also contains glossy materials. BIS works well on the killeroo's lower back (detail 2, variance due to BRDF variation) but poorly on the neck (detail 1, variance due to visibility variation). IC performs in the

opposite way. The highly glossy BRDF and complex curvature on the back invalidate the distributions stored at sparse importance records. Our method is much less sensitive to geometric and material variations and significantly reduces noise on both the back and neck.

The next two scenes, *Sponza* and *Room*, were designed to experiment with indirect lighting. *Sponza* (Figure 3.12) was rendered with an environment map (approximated with 8K lights) and 70K indirect VPLs. In this scene, only a small number of VPLs are visible to any shading point. BIS is ineffective since it can not locate those visible lights which make contributions to the shading point. In this scene, our method achieves great improvement compared to BIS, but is slightly less effective than IC because we only use cluster-level importance. Lights within a cluster are still sampled uniformly, rather than by their importance. On the other hand, IC computes individual contributions for each light at every importance record. Thus, when rendering flat regions without much visibility variation, IC generates less noise than VsibilityCluster (detail 1). However, IC renders the end of the alley less effectively (detail 2), where it lacks importance records. For the *Room* scene (Figure 3.13), IC produces noises on the boundary of the two dragons where the complex geometry invalidates the importance records (detail 1). Our method, in contrast, can better cope with glossy materials and geometry complexity.

The last two scenes, *Construction* (Figure 3.14) and *Hairball* (Figure 3.15), demonstrate situations with complex illumination and geometry. In *Construction*, there are lots of surrounding lattices, handrails, and trestles, resulting in large visibility variations. The lights coming from the right side are blocked by a group of pillars and shine through the gaps to a bump mapped ground. For this complex scene, our method produces noticeable noise but still outperforms the other two methods by a margin. In *Hairball*, the complex self-occlusion makes BIS and IC very ineffective. It is worth noting that IC does not work better than BIS in this scene because of the highly complex geometry.

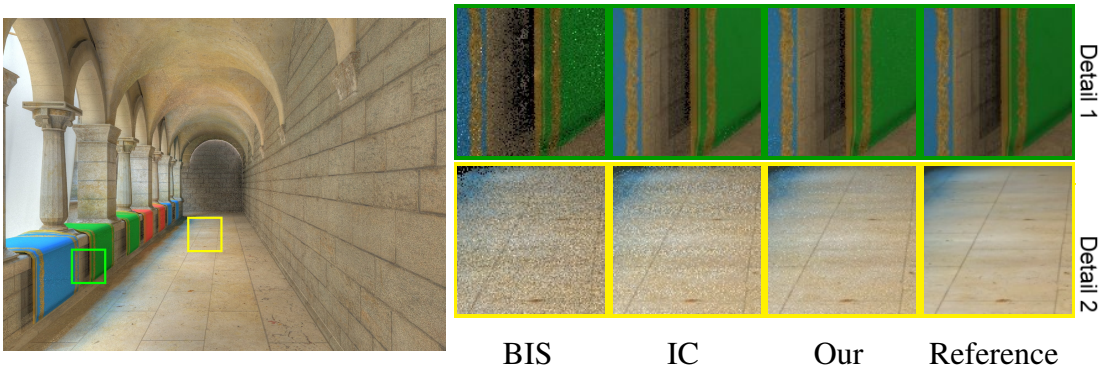


Figure 3.12: *Sponza* (equal-time comparison, 600 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. IC achieves the best noise reduction in flat regions (detail 1), but is less effective for regions that are far away from the camera (detail 2) due to the insufficient caches. Our method reduces noise consistently across the whole image.

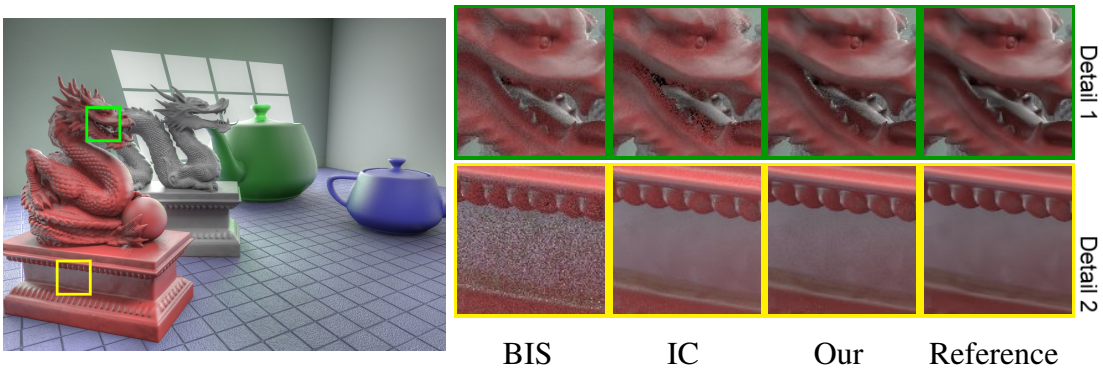


Figure 3.13: *Room* (equal-time comparison, 300 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. Because dragons have fine geometry details, IC's caches are less accurate (detail 1). Our method exploits geometry coherence within shading point clusters and handles scenes with fine geometry details better.

Figure 3.16 visualizes errors, and lists the mean squared error (MSE) for all compared methods and scenes. In most scenes (excepts for *Sponza*), our method has significantly lower MSE than BIS and IC. IC is slightly better than our method for *Sponza* because the scene contains mostly flat surfaces.

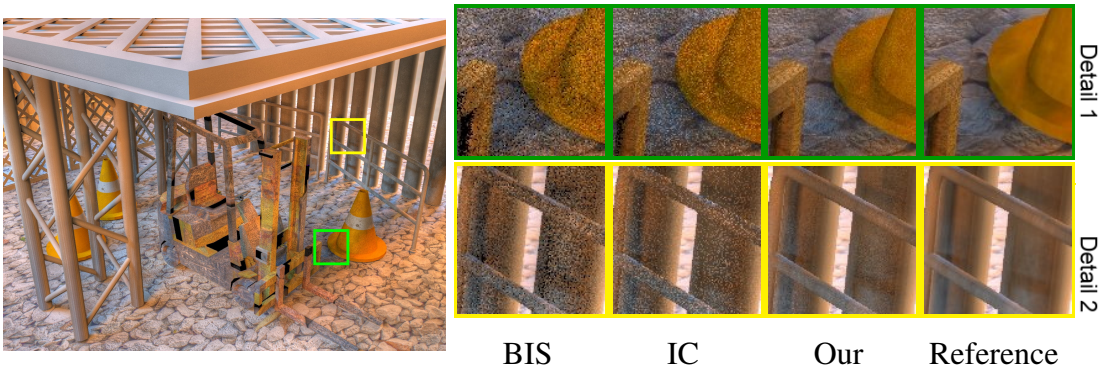


Figure 3.14: *Construction* (equal-time comparison, 500 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. The grating patterns produced by pillars and the thin geometry such as lattices, handrails and trestles result in high-frequency visibility variation, making BIS and IC very inefficient.

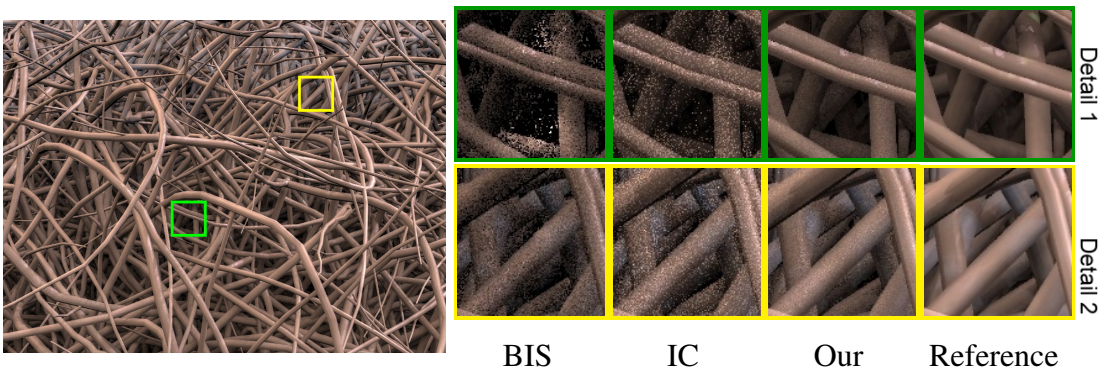


Figure 3.15: *Hairball* (equal-time comparison, 800 seconds). Left: full image rendered by our approach. From column 2 to column 5: detailed images of BIS [109], IC [45], our approach, and the reference. The complex geometry results in large visibility variation, causing relatively large noise in all methods. However, since VisibilityClusters are refined for each shading cluster, it achieves superior noise reduction compared to BIS and IC.

Discussions

BIS generally converges fast in unoccluded regions with a few samples, but becomes inefficient where there is occlusion. The worst case occurs when the directions of strong lights or BRDF peaks are occluded. In this circumstance, BIS becomes very inefficient because most samples are wasted on directions without contributions.

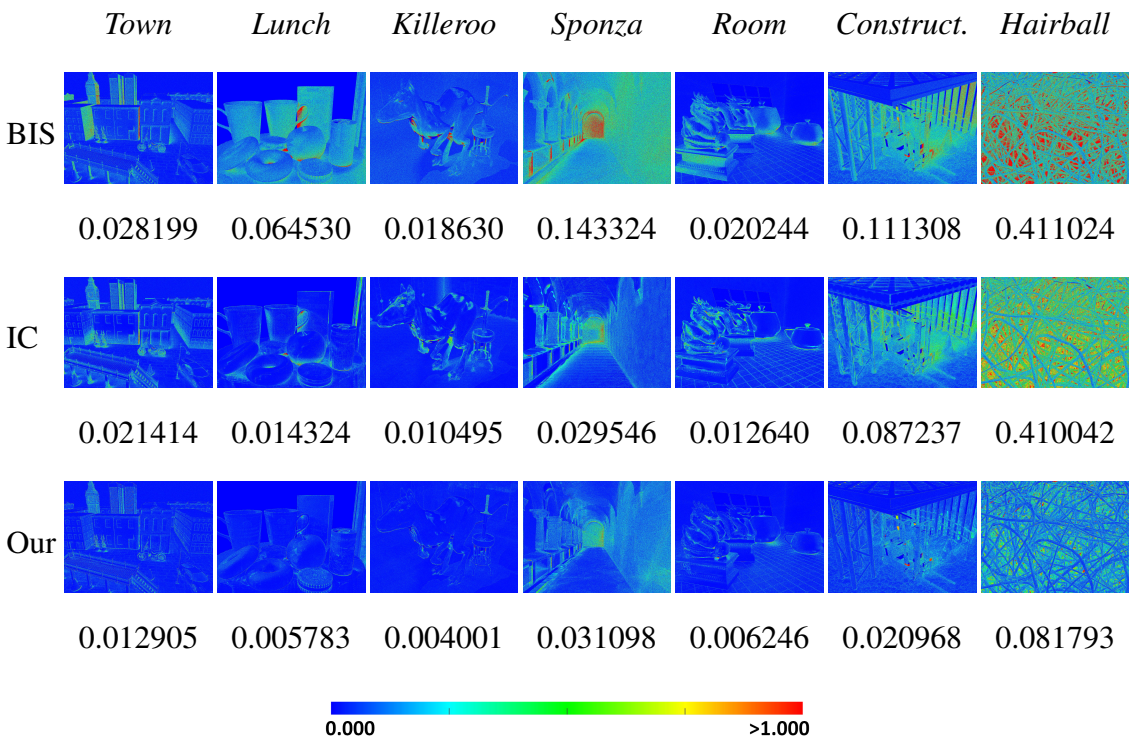


Figure 3.16: Visualization of errors. Rows from top to bottom: BIS [109], IC [45], and our approach. The value under each image is its MSE.

IC greatly alleviates variance in areas with smooth surfaces. However, for more complex surfaces and glossy materials, the uniform distribution of importance records in the image plane could miss fine details or scene features, resulting in less effective variance reduction in these areas. Finally, it does not scale well to the number of VPLs. Since the full contributions need to be evaluated and stored for every importance record, the time of drawing samples and the memory requirements quickly grow as the number of VPLs increases.

VisibilityCluster significantly reduces variance for scenes with different illumination conditions and complex geometry. Its scalability to the geometric complexity is improved by grouping similar shading points; similarly, clustering lights and only estimating average visibility improve the scalability in the number of virtual point lights.

Limitations

The *Sponza* scene (Figure 3.12) reveals a limitation of our method. In this scene, very few lights are visible for a shading point, making our cluster-level importance sampling less effective. The same problem also appears inside the forklift in the *Construction* scene (Figure 3.16). Another limitation is that in very simple scenes with little occlusion, the overhead of computing VisibilityCluster might not justify its gain.

Memory usage

The memory usage required for VisibilityCluster is $N_{SC} \times N_{LC} \times 4$ bytes, where N_{SC} and N_{LC} are the number of shading clusters and light clusters, respectively. For 6,400 shading clusters and 1,200 light clusters (the average case in this paper), it uses about 30 MB. The construction of the shading tree and the light tree requires additional memory, but it can be released once the clusters are determined. Compared to IC, our memory requirement is significantly smaller.

3.5.2 Directional occlusion

VisibilityCluster can also be used to approximate directional occlusion for lighting previews. A row in the VisibilityCluster matrix (for a shading point) gives the average visibility of each light cluster with respect to the shading point. It can be used as an approximation of directional occlusion. In this setting, we approximate the reflected radiance in Equation 3.2:

$$L \approx \sum_{L_s} \tilde{L}_i(L_s) \bar{f}_r(L_s) \bar{V}(L_s), \tag{3.7}$$

where L_s is a light cluster; $\tilde{L}_i(L_s)$ is its total illuminance; $\bar{f}_r(L_s)$ is the average BRDF from the light cluster to the shading point and $\bar{V}(L_s)$ is the average visibility between the light cluster and the shading point. Our method can be used for estimating $\bar{V}$.

(a) Local shading with 1,200 representative lights (centres of light clusters), 35.4 seconds

(b) Reference image with 32K lights, 10,047.0 seconds

(c) Approximated DO with 1,200 light clusters using VisibilityCluster directly, 47.8 seconds

(d) Approximated DO with 1,200 light clusters using interpolated VisibilityCluster, 59.5 seconds

Figure 3.17: The *Statue* scene illuminated by 32K lights. (a) Local shading with 1,200 representative lights without considering visibility. (b) Reference image that accumulates contributions of all 32K lights with visibility included. (c, d) Lighting previews using VisibilityCluster visibility and interpolated VisibilityCluster visibility for directional occlusion (DO). Directly using VisibilityCluster results in blocking artifacts (c), which are alleviated by combining visibility terms of nearby VisibilityClusters (d).

Figure 3.17 presents an example for enhancing local shading using VisibilityCluster. Figure 3.17(a) gives the result of local shading without considering visibility to lights. It looks quite different from the reference (Figure 3.17(b)) in both, shadows and tone. By incorporating the approximated visibility supplied by VisibilityCluster, Figure 3.17(c) significantly improves the shadows and tone. However, since average visibility is com-

puted per VisibilityCluster, directly using it for $\bar{V}$ could result in blocking artifacts (Figure 3.17(c)). The problem can be alleviated by blending visibility terms of nearby VisibilityClusters. A kd-tree is built for the centres of all shading clusters. To determine $\bar{V}(L_s)$ for a shading point, we look up the n nearest shading clusters based on geometric properties in the kd-tree, and linearly blend their $\bar{V}(L_s)$ values using the reciprocals of distances as weights. We used $n = 3$ in our implementation. As Figure 3.17(d) shows, although mild blocking artifacts are still visible, the directional shadows on the floor look better, and surface details on the statue are also greatly improved. With very little cost, by adding directional occlusion, VisibilityCluster significantly enhances the quality of local shading.

3.6 Conclusion and Future Work

The VisibilityCluster we introduced in this chapter is an efficient estimation and compact representation for averaged shading point visibility. By properly clustering shading points and lights, visibility entries in the many-light transport matrix exhibit a block-like structure and can be efficiently approximated by sampling. The submatrix formed by a cluster of shading points and a cluster of lights is called a VisibilityCluster. We demonstrate that the average visibility of a VisibilityCluster can be faithfully estimated with only few shadow tests. We also use the average visibility as an estimation of quality to guide further refinement of clustering for improving accuracy. The result is an efficient algorithm for estimating visibility that can be used in applications such as importance sampling and directional occlusion.

One interesting future direction is to extend our method to the temporal domain for rendering animations. We are planning to investigate possible exploitation of temporal coherence to reduce per-frame cost. Another possible direction is to deal with glossy materials. It is interesting to employ advanced VPL methods that focus on glossiness [53, 29]. Finally, we would also like to combine our method with some recently proposed ap-

proaches for further speeding up visibility estimation, e.g. using RTSAH [55] to accelerate shadow tests.

Chapter 4

Dual-Matrix Sampling for Scalable Translucent Material Rendering

In previous chapters, we assume that lights can only be reflected when they interact with surfaces. Such assumption fails for translucent materials which allow lights to enter and scatter within the media, a process called subsurface scattering. Simulating subsurface scattering can greatly increase visual richness, however, at the cost of significant computation. Translucent material rendering is especially time-consuming in large scenes with complex illumination because each path is very expensive. This chapter introduces a scalable algorithm for rendering translucent materials with complex lighting. We represent the light transport of a diffusion approximation by a dual-matrix representation with Light-to-Surface and Surface-to-Camera matrices. By exploiting the structures within the matrices, the proposed method can locate surface samples with little contribution by using only subsampled matrices and avoid wasting computation on these samples. The decoupled estimation of irradiance and diffuse BSSRDFs also allows us to have a tight error bound, making the adaptive diffusion approximation more efficient and accurate. Experiments show that our method outperforms previous methods for translucent material rendering, especially in large scenes with massive translucent surfaces shaded by complex illumination.

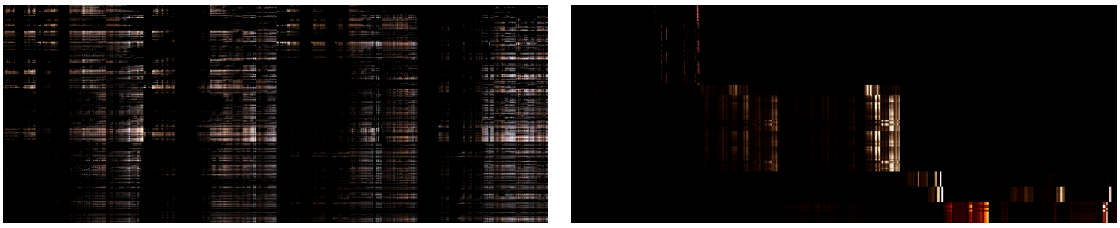
(a) *Sculpture* (b) *Exhibition* (c) *Cathedral* (d) *Room*

Figure 4.1: Several scenes which are inefficient to render with previous methods [57, 4]. In (a) and (b), only a small part of the translucent surfaces show up in the final image due to a close-up viewing or occlusion. The traditional two-pass method [57] which computes surface irradiance in a view-independent manner would waste computation on unimportant surface samples. The single-pass approach based on lightcuts [4] alleviates this problem but does not work well for cases with backlit objects (c) or difficult light paths (d).

4.1 Overview

Translucent materials, such as jade, marble, milk, skin, meat, and leaves, are very common in real world. Correctly modeling and rendering their appearance is very important for realistic rendering. Different from surface reflection, light could enter such materials and scatter within them. Such subsurface scattering gives these materials distinct soft look. However, its simulation with Monte Carlo method is computationally expensive because of the needs for tracing a large number of long complex paths especially for highly scattering materials. The analytical dipole diffusion model [58] and later improved models [32, 31] provide efficient solutions for multiple scattering and have been widely adapted in industry and film production.

For efficiency, these diffusion models are usually evaluated with the two-pass approach proposed by Jensen and Buhler [57]. In the first pass, the irradiance values at a set of selected surface samples are precomputed and stored in a hierarchical structure. These pre-computed irradiance values are then adaptively gathered with the diffusion kernel at



(a) Light-to-Surface (b) Surface-to-Camera

Figure 4.2: Matrix visualization of the ChessGame scene in Figure 4.9. As the figures show, the Light-to-Surface matrix has a low-rank structure while the Surface-to-Camera matrix is very sparse.

camera samples in the second pass. Although possessing the advantages of reusing pre-computed irradiance, this approach still suffers from the problem that computation could be largely wasted on the unimportant surface samples which either are occluded by other objects or contribute very little to the final image. As a result, the precomputation of surface irradiance samples in the first pass often becomes the performance bottleneck, especially in a large scene with massive translucent objects shaded by complex illumination. Figure 4.1(a) and 4.1(b) demonstrate two such examples: close-up rendering of a sculpture and a large exhibition with difficult occlusion. Arbree *et al.* [4] proposed an alternative approach to address this issue. Based on the lightcuts framework [107, 106], their method performs adaptive clustering on light-surface-camera paths. Surface irradiance values are computed on demand by bounding the errors of path clusters. However, because visibility is not taken into account during the error estimation, errors could be over-estimated with their conservative bound when bright lights are occluded. This makes their approach less effective for scenes with backlit translucent objects (Figure 4.1(c)) or difficult light paths (Figure 4.1(d)). Moreover, their single-pass strategy also makes the reuse of surface irradiance less intuitive and complicates the implementation.

We therefore introduce an efficient method for scalable translucent material rendering, possessing the advantages of reusing pre-computed irradiance while largely reducing

computation on unimportant surface irradiance samples. Inspired by recent matrix sampling and reconstruction methods [54, 80] (where light transport in the many-light setting is represented by a Light-to-Camera matrix), we decompose the light transport tensor (light-surface-camera) for translucent material rendering into two sub-matrices, the Light-to-Surface matrix and the Surface-to-Camera matrix. Figure 4.2 shows the Light-to-Surface and Surface-to-Camera matrices of the ChessGame scene (partial, Figure 4.9). We observe that, by clustering properly, the Light-to-Surface matrix exhibits a good coherence structure with the property of a low rank. The Surface-to-Camera matrix is very sparse. For each camera sample, there are few significant terms and these terms often group together. Our algorithm exploits the specific structures of both matrices for estimating the surface irradiance values and diffuse BSSRDFs adaptively. The advantages of our method are twofold. First, it is more efficient. The combined information from both matrices helps predict which surface samples would be less important during the adaptive diffusion gathering step. These samples can be largely approximated and calculated more quickly. Second, it is more accurate. The decoupled estimation of irradiance values and diffuse BSSRDFs allows us to have a tight error bound. The tight bound not only makes the prediction of surface importance more accurate, but also the adaptive diffusion approximation more efficient. Experiments show that our method significantly outperforms previous methods on a set of complex scenes with massive translucent objects and complex lighting.

The rest of the chapter is organized as follows. Section 4.2 discusses related work. Section 4.3 describes the procedure of dual-matrix sampling. Experiments are described in Section 4.4. Finally, Section 4.5 concludes the chapter with directions for future work.

4.2 Related Work

In this section we briefly review previous work on modeling and rendering for subsurface scattering, with an emphasis on methods using diffusion approximation. Methods for translucent material acquisition and editing are out of the scope of this dissertation and not included here.

4.2.1 BSSRDF models for subsurface scattering

Subsurface scattering is usually modeled by the Bidirectional Subsurface Scattering Reflectance Distribution Function (BSSRDF) [77]. To compute the outgoing radiance L reflected at a point x along direction ω_o, we need to integrate the product of the BSSRDF S and the incoming radiance over the surfaces of translucent objects:

$$L(x,\omega_o) = \int_A \int_\Omega S(x_o,\omega_o;x_i,\omega_i) L(x_i,\omega_i)(n\cdot\omega_i)d\omega_i dA(x_i), \tag{4.1}$$

where $L(x_i,\omega_i)$ is the incident radiance at a point x_i along the incoming direction ω_i. Jensen *et al.* [58] split the BSSRDF into a single scattering term S^1 and a multiple scattering term S^d:

$$S(x,\omega_o;x_i,\omega_i) = S^1(x,\omega_o;x_i,\omega_i) + S^d(x,\omega_o;x_i,\omega_i). \tag{4.2}$$

The expensive multiple scattering term S^d can be efficiently approximated with the dipole source approximation and solved analytically with a diffuse BSSRDF R_d. The outgoing radiance due to the multiple scattering term is therefore approximated as:

$$\begin{aligned} L^d(x,\omega_o) = \frac{1}{\pi} F_t(\eta,\omega_o) \int_A R_d(\|x_o - x_i\|)\cdot \\ \int_\Omega F_t(\eta,\omega_i) L(x_i,\omega_i)(n\cdot\omega_i)d\omega_i dA(x_i), \end{aligned} \tag{4.3}$$

where F_t is the Fresnel transmittance and η is the relative index of refraction.

Later, several models have been proposed for improving the dipole model. Donner and Jensen [36] used the multipole model for rendering multi-layer transparent materials.

Later, they also proposed a hybrid model with photon diffusion to account for occlusion and caustics inside the medium [37]. Recently, quantized diffusion (QD) [32] and its analytical version, the better dipole [31], have been proposed for improving the accuracy of dipole models. To further improve QD, Yan *et al.* [112] made correction of oblique illumination. Habel *et al.* [48] also proposed a correction factor to correct the overestimation of near-surface sources.

4.2.2 Subsurface scattering rendering

One of the most popular approach for rendering with the dipole model is the two-pass method proposed by Jensen and Buhler [57]. In their approach, irradiance is first computed at surface samples and then adaptively gathered with a hierarchical structure at each camera sample. The method robustly generates smooth images by reusing irradiance values from a fixed set of surface samples.

An alternative for integrating over surface is to generate samples in the surroundings of camera samples on demand [108, 62]. This kind of methods do not require additional memory for storing surface samples, however, at the expense of less intuitive reuse of surface irradiance.

Caching-based approaches have also been proposed. Inspired by irradiance caching, Keng *et al.* [61] proposed to cache subsurface scattering in image space and perform interpolation based on gradients. The multiresolution radiosity caching [19] instead caches irradiance values on vertices of micropolygons, which are further reused by nearby camera samples.

Some approaches combine the dipole approximation with Monte Carlo simulation for more accurate rendering [17, 70, 102]. These methods share the same spirit by splitting the volume into parts, the inner core and the outer shell. The inner part is rendered using the dipole solution while the outer one is simulated using Monte Carlo method.

In the context of interactive translucent materials rendering, lots of approaches have been proposed to exploit the graphics hardware. Radiosity-based methods achieve high frame-rate by pre-computing the diffusion transport between vertices or patches [69, 14, 51]. Other approaches avoid pre-computation by first rendering irradiance map into textures and later sampling or filtering it [25, 6, 47, 71, 15, 90]. d'Eon *et al.* [33] extended translucent shadow maps [25] and texture-space diffusion for rendering human skin. Munoz *et al.* [74] modeled the subsurface scattering as image convolution and solved it using discrete Fourier transform. These approaches, however, either restrict illumination to simple light sources or are not scalable to complex scenes.

Other methods render translucent materials under complex illumination in real-time using expensive precomputation [94, 110, 95]. The approach proposed by Sheng *et al.* [91] is also based on precomputation but focuses on the inter-reflection between translucent objects.

With the same goal as our method, Arbree *et al.* [4] proposed a scalable single-pass method to address the excessive irradaince computation of the two-pass method [57] based on the lightcuts framework. Surface irradiance is computed adaptively until the contribution is smaller than an estimated error bound. However, as the original lightcuts approach, their method becomes less effective when complex occlusion occurs. Another disadvantage of their single-pass strategy is that it requires form-factor caches to reuse surface irradiance samples. It not only complicates the implementation, but also becomes ineffective in complex scenes because the cache hit rate could be extremely low, as reported in Arbree's thesis (e.g. 1%) [3].

4.3 Dual-Matrix Representation and Sampling

4.3.1 Algorithm Overview

As stated in Section 4.1, in translucent material rendering, the contribution of a camera sample can be calculated by integrating irradiance values of surface samples with distance-dependent diffusion kernels, while the irradiance of a surface sample is calculated by summing contributions from light samples. Thus, in principle, to use the matrix sampling approach, the light transport in our setting can be modeled by a 3D Light-Surface-Camera tensor. However, we found that the tensor-based approach consumes more memory and could be redundant in our application. Thus, we decompose the light transport into two matrices for Light-to-Surface and Surface-to-Camera light transport.

To use the matrix representation, we first discretize the lights, surfaces, and final image into point samples. Light samples are generated as in previous many-light approaches [107, 54]. Surface samples are created by using a ray-tracing-based Poisson-disk sampling approach [68]. Camera samples are 3D intersections of the scene and eye rays corresponding to sub-pixel samples for anti-aliasing.

Our light transport representation consists of Light-to-Surface and Surface-to-Camera matrices (Figure 4.2). The Light-to-Surface matrix describes the energy transport from light samples (columns) to surface samples (rows). Matrix entries are lighting contributions. By summing along each row, we obtain the irradiance value of each surface sample. Similarly, the Surface-to-Camera matrix describes the energy transport from surface samples (columns) to camera samples (rows). Matrix entries are diffusion contributions. The pixel colors of the final image can be obtained by multiplying the surface irradiance vector with the Surface-to-Camera matrix. As demonstrated in Figure 4.2, because the characteristics of light transport and diffusion transport differ, these two matrices exhibit distinct structures. The Light-to-Surface matrix has both global and local structures and

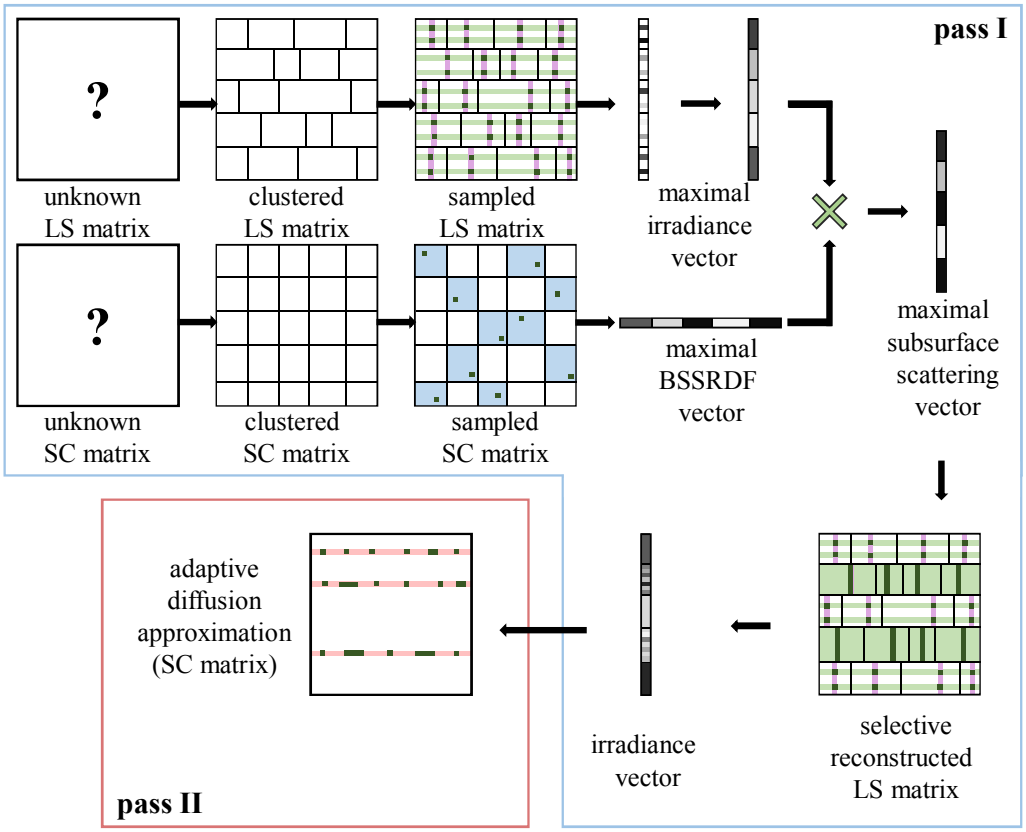


Figure 4.3: The flowchart of our algorithm. Our method has two passes. In the first pass, we estimate the maximum contribution of each surface cluster. This is done efficiently by exploiting the matrix structures in Light-to-Surface (LS) matrix (for irradiance) and Surface-to-Camera (SC) matrix. Only the matrix entries labelled in dark green are computed. Based on the estimated maximum contribution, we then selectively reconstruct surface irradiance. Surface samples which have little contributions to the final image will be replaced by a single cluster representative. In the second pass, our adaptive diffusion approximation is based on error, instead of heuristic refinement in traditional two-pass approach [57].

the Surface-to-Camera matrix tends to be more local. Our algorithm exploits these structures to cluster the surface samples with little contribution, avoiding wasting computation resources on unimportant surface samples to the final image.

Figure 4.3 demonstrates the flowchart of our algorithm. There are two passes. The first pass generates the surface irradiance vector while the second pass computes the final image. Since not all surface samples make significant contributions to the final image,

we do not need to compute a fully accurate surface irradiance vector. We exploit the matrix structure for locating important surface samples. In the first pass, we first coarsely approximate the Light-to-Surface matrix and the Surface-to-Camera matrix to estimate the irradiance and diffusion reflectance of surface samples, respectively. Due to the local structures of both matrices, we can accurately predict how much a surface samples would influence the camera samples by sampling the low-resolution matrices. This allows us to selectively reconstruct the Light-to-Surface matrix and obtain the surface irradiance vector with sufficient precision. The second pass renders the final image by adaptively reconstructing the Surface-to-Camera matrix. Since the surface irradiance has been obtained in the first pass, the potential error for adaptive diffusion gathering can be bounded tightly.

4.3.2 Dual-matrix Sampling

Light-to-Surface matrix reconstruction

The goal of the first pass is to obtain the surface irradiance vector. To avoid overspending computation on unimportant surface samples, we need to identify them without computing them first. As seen in Figure 4.2, because a cluster of nearby samples usually have similar behaviors, both transport matrices are low-rank in nature. Thus, their subsampled versions provide very good approximations to the full versions if properly clustered. For clustering surface samples, we first build a bounding volume hierarchy. Clustering is determined by finding a cut in the hierarchy of surface samples. Starting from the root, we continue to replace the node with the maximum extent with its two children until the desired number of clusters has been reached. For clustering light and camera samples, we use the same method used in LightSlice [80].

The contribution of a surface sample depends both its irradiance value and diffuse BSSRDF value. As stated in the previous paragraph, it would be sufficient to estimate

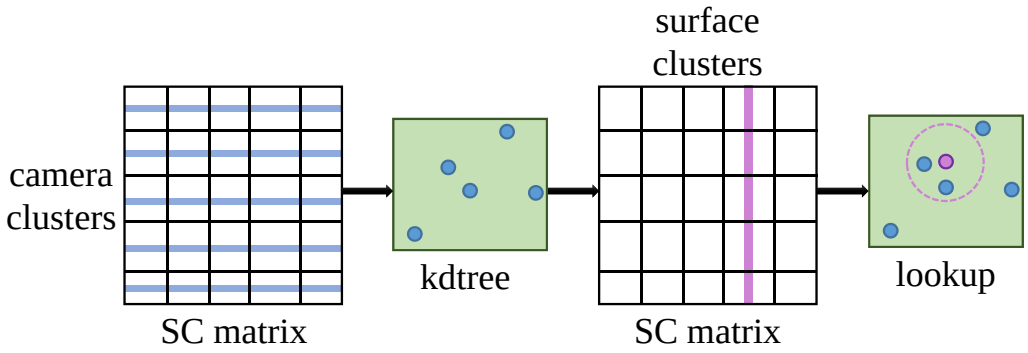


Figure 4.4: Algorithm for Rd estimation. We first build a kdtree for the centers of all camera clusters. Then for each surface cluster, we lookup its K nearest neighbors in the kdtree. The maximum Rd of a surface cluster is calculated by taking the maximum Rd values between the surface cluster and the K camera clusters.

the contribution at the cluster level. As we will introduce in Section 4.3.2, the maximal contribution of a surface cluster can be estimated by the product of its maximal irradiance and diffuse BSSRDF values. We can estimate the former from the Light-to-Surface matrix and the latter from the Surface-to-Camera matrix.

The structure of the Light-to-Surface matrix is similar to the Light-to-Camera matrix in traditional many-light approaches. The only difference is that some surface samples might not contribute to the final image. Since the matrix has been shown locally low-rank [54, 80], we can obtain an accurate approximation with a small set of representative lights and surface samples. The detailed steps are described as follows: We first randomly select a few surface samples in each surface cluster (rows labelled in light-green in Figure 4.3). Next, we use LightSlice [80] to select a small set of representative lights for each cluster (columns labelled in purple in Figure 4.3). The irradiance values of the selected samples are then evaluated using these representative lights. The result is a low-resolution Light-to-Surface matrix with a small number of surface and light samples (matrix entries labelled in dark-green in Figure 4.3). By taking row sums, we obtain a reduced surface irradiance vector. Since our goal is to obtain the maximal irradiance for each cluster of surface samples, we take the maximum operator for each cluster.

The structure of the Surface-to-Camera matrix is even more sparse after proper clustering. This means only a few camera clusters receive contributions from a surface cluster. Since the diffusion kernel depends on the distance, camera clusters close to a surface cluster are more likely to receive its contribution. Based on this observation, we estimate the upper bound of diffuse BSSRDF of a surface cluster from its nearby camera clusters. Figure 4.4 illustrates the procedure. We first cluster the camera samples according to their spatial positions. Next, a kd-tree is built for all centers of camera clusters. For each surface cluster, we find its K nearest camera clusters (K is usually set to 10 in our implementation). For each camera cluster, we estimate their R_d using the shortest distance between the bounding volumes of both clusters. The maximum of the K R_d values is returned as the estimation of the maximal R_d of the surface cluster.

Finally, we obtain the estimated maximum contribution of a surface cluster by multiplying its maximum irradiance and R_d. If the contribution is too small (smaller than the acceptable error ε), we simply average the estimated irradiance values of the selected samples for the surface cluster and use it for all surface samples in the cluster. For other surface clusters, they are likely to have significant contributions and need to be evaluated more accurately. Thus, for each samples in the cluster, we use LightSlice [80] to compute its irradiance. Through the process, we obtain the surface irradiance vector which have accurate values for important surface samples and rough estimations for unimportant ones. It is worth mentioning that in our framework LightSlice can be replaced with any other light importance sampling algorithms [54, 45].

Please note that our estimation of maximum irradiance is only an approximation and could be smaller than the real maximum since we use sparse samples. However, in practice, we found that it has little influence on the rendered results. It is similar to the argument made by Delves and Mohamed [30], in which they argue that a realistic error estimation is preferred over a pessimistic bound.

Surface-to-Camera matrix reconstruction

With the surface irradiance vector obtained from pass 1, pass 2 reconstructs the Surface-to-Camera matrix and renders the image by adaptively refining a hierarchy of surface samples in a similar way as previous work [57]. We reuse the bounding volume hierarchy of surface samples built in previous pass. The interior node defines a cluster of surface samples. For computing the color of a camera sample, we traverse the hierarchy to approximate the diffusion transport from surface samples based on the following error metrics:

$$\left\| L_C^{true} - L_C^{est} \right\| \leq R_d^{(ub)} \left[\sum_{j \in C} E_j A_j \right], \tag{4.4}$$

where $R_d^{(ub)}$ is the upper bound of diffusion transport from the camera sample to the surface cluster C determined by computing the shortest distance from the camera sample to the bounding volume of node C; $E_C^{(max)}$ is the maximum irradiance value in node C; and A_j is the area of a surface sample j in C.

Equation 4.4 gives the maximal contribution of the surface cluster C to the camera sample. Since the error of estimating a cluster's contribution with its representative sample cannot be larger than its maximal contribution, Equation 4.4 also gives a bound on the error of adaptive approximation. We repeat substituting the node C with its two children if the maximum error is larger than a predefined acceptable error ε. The color of the camera sample is estimated by summing contributions from the resulting configuration of surface clusters.

It is worth noting that the Equation 4.4 is only bounded by the maximum diffuse BSSRDF $R_d^{(ub)}$ because we already know the irradiance of surface samples in C. The dual-matrix representation allows us to obtain a tighter error bound than previous approaches.

Comparison to traditional two-pass approach. The two-pass approach proposed by Jensen and Buhler [57] adopted a heuristic error metric based on the approximated

maximum solid angle that a surface cluster C covers toward a camera sample x_o:

$$\Delta\omega = \frac{A_C}{\|x_o - P_C\|}, \tag{4.5}$$

where A_C is the total surface area in node C and P_C is the position of surface representative, computed by averaging the position of surface samples in the cluster weighted by their irradiance values. For a given camera sample, this error metric spends more effort on refining nearby surface clusters, with little consideration of surface irradiance. When the lighting is non-uniform (for example, strongly backlit), it requires more time to converge.

Comparison to the single-pass approach. Arbree *et al.* [4] estimate the error bound of irradiance based the lightcuts [107, 106] and combine it with the bound of diffuse BSSRDF, resulting in:

$$\left\|L_C^{true} - L_C^{est}\right\| \le R_d^{(ub)} F^{(ub)} \left[\sum_{i \in C_L} I_i\right]\left[\sum_{j \in C_B} A_j\right]. \tag{4.6}$$

Since the irradiance is computed on demand during the adaptive refinement. Its value is unknown and can only be conservatively estimated by the upper bound of form factor $F^{(ub)}$ multiplied by total intensity from lights $\sum_{i \in C_L} I_i$. The negligence of visibility is the main weakness of this metric.

Figure 4.5 demonstrates the images rendered with two metrics, maximum solid angle approximation (MSA) in traditional two-pass method [57] and ours. Although the two methods converge to the correct results finally, under limited time budget, our method can locate the surface samples with potentially large error and refine them first. This benefits in applications like lighting previews.

4.4 Experiments

We implemented the proposed algorithm on top of the PBRT2 system [82]. All results were generated on a machine with an Intel Xeon E5-2650 CPU at 2.0 GHz, 128GB of RAM, and using 32 threads.

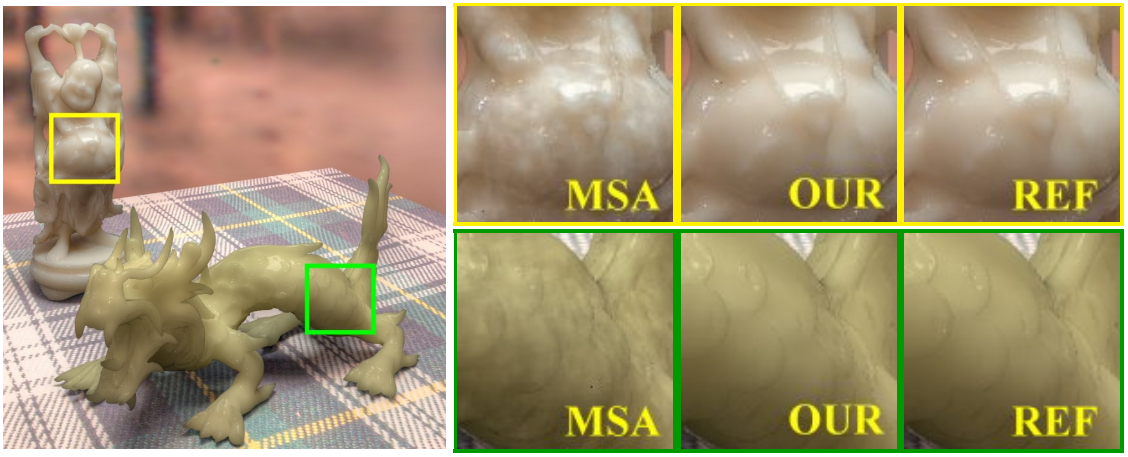
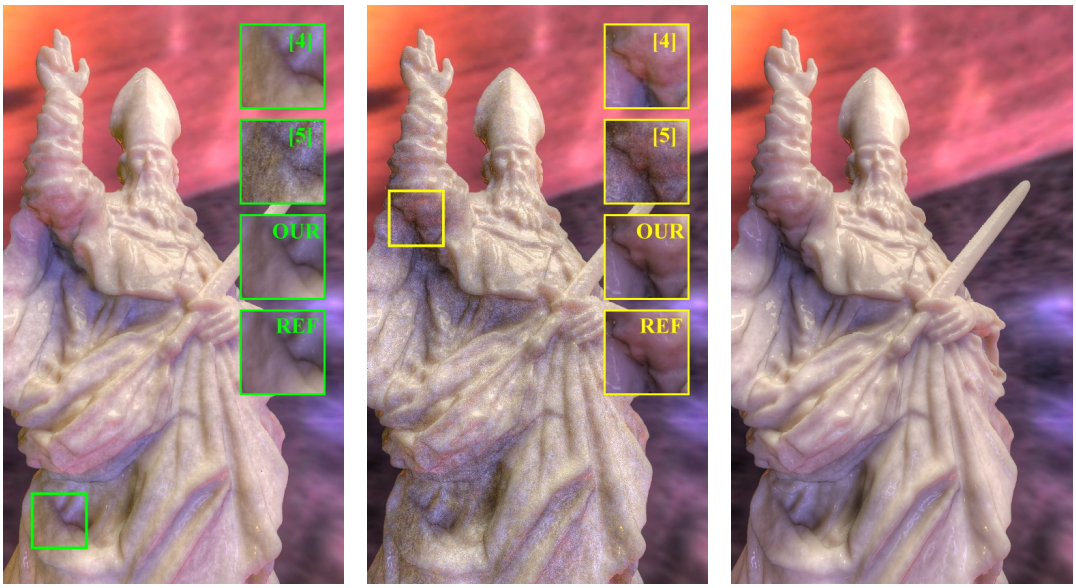


Figure 4.5: Equal-time comparison (50 sec.) of the refining metrics: maximum solid angle, MSA (Equation 4.5), and our approach (Equation 4.4). Our method can locate surface samples with larger contribution, making the adaptive diffusion approximation more efficiently.

We design five scenes, with different illumination conditions and geometry complexities, to access the performance of our algorithm. The first two scenes, *Sculpture* (Figure 4.6) and Cathedral (Figure 4.7), have relatively simple layouts. In *Sculpture* scene, a marble stone carving is illuminated by a high-frequency environment map and a large area light. The scene demonstrates a case of close-up rendering. Only a small part of the sculpture is inside in the view frustum. In *Cathedral* scene, the jade-made lucy model is placed in front of a stained glass (modeled as an environment map). We use this scene to experiment with back lighting. The remaining three scenes are more complex. The *Room* scene (Figure 4.8) contains very difficult light paths. The illumination of this scene comes from an environment map and an indoor area light. Lights from the environment map can only come into the room though the door and only their indirect bounces can shade the translucent objects on the desk. The *ChessGame* scene (Figure 4.9) contains a large number of translucent objects: two kinds of chesses, a vase, a set of teapot and cups, and a candle. Illumination comes from a high-frequency environment map and an area light. The last scene, *Exhibition* (Figure 4.10), contains massive translucent statues.



(a) Jensen and Buhler [57] (b) Arbree et al. [4] (c) Our method

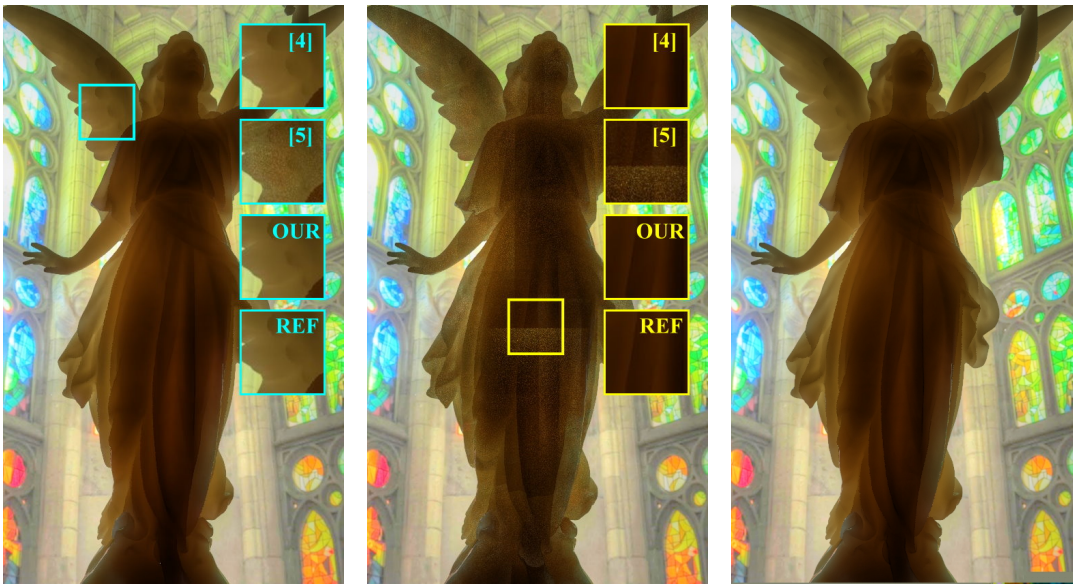
Figure 4.6: Equal-time comparison of the *Sculpture* scene (30 sec.). The RMSE values of images (a) to (c) are 0.003270, 0.003040, and 0.000588, respectively.

The scene is illuminated by an environment map and five area lights. Due to the complex layout, only a small part of translucent surfaces appear in the final image.

All the parameters of translucent materials in these scenes are based on previous measurement [58, 75] and rendered with the better dipole model [31]. The *Sculpture* and *Cathedral* scene were rendered at a resolution 500 x 850 and the *Room*, *ChessGame*, and *Exhibition* scene were rendered at a resolution 800 x 600, all with 16 sub-pixel samples for antialiasing (for efficiency, subsurface scattering is computed one time per pixel). Finally, surfaces without subsurface scattering were rendered with the matrix row-column sampling approach [54]. Other details of the five scenes are summarized in table 4.1.

We conduct comparisons with the following approaches:

- Jensen and Buhler [57]: the traditional two-pass algorithm which computes surface irradiance in a brute-force manner and adopts maximum solid angle approximation (Equation 4.5) for adaptive diffusion approximation. We follow the implementation



(a) Jensen and Buhler [57] (b) Arbree et al. [4] (c) Our method

Figure 4.7: Equal-time comparison of the *Cathedral scene* (30 sec.). The RMSE values of images (a) to (c) are 0.000107, 0.000330, and 0.000082, respectively.

Scene	# Triangles	# VPLs (Type)	# Surface Samples
Sculpture	800,179	18,432 (Environment map)	13,026,444
Cathedral	157,740	11,741 (Environment map)	6,565,679
Room	503,741	165,912 (Global illumination)	13,761,152
ChessGame	1,514,008	52,675 (Global illumination)	11,202,462
Exhibition	7,354,427	186,448 (Global illumination)	20,512,386

Table 4.1: Statistics of the five test scenes. The light samples (VPLs), including both direct and indirect illumination, were obtained by uniform sampling the environment light and tracing photons. The surface samples were uniformly distributed on all translucent surfaces (objects that fully outside the view frustum were culled).

provided by PBRT [82]. Matrix sampling approaches [54, 80] are employed for reducing the number of light samples in irradiance computation.

- Arbree et al. [4]: the single-pass approach based on lightcuts. We implement their method upon PBRT [82]. The size of form-factor cache is set to 100,000. All other parameters are set according to the authors' suggestion.

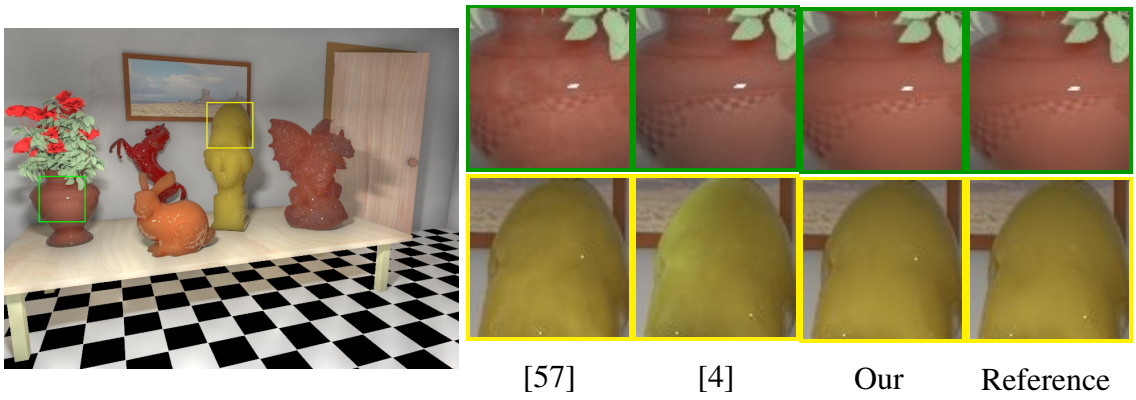


Figure 4.8: Equal-time comparison of the *Room* scene (300 sec.). The RMSE values of images (a) to (c) are 0.000347, 0.001420, and 0.000114, respectively.

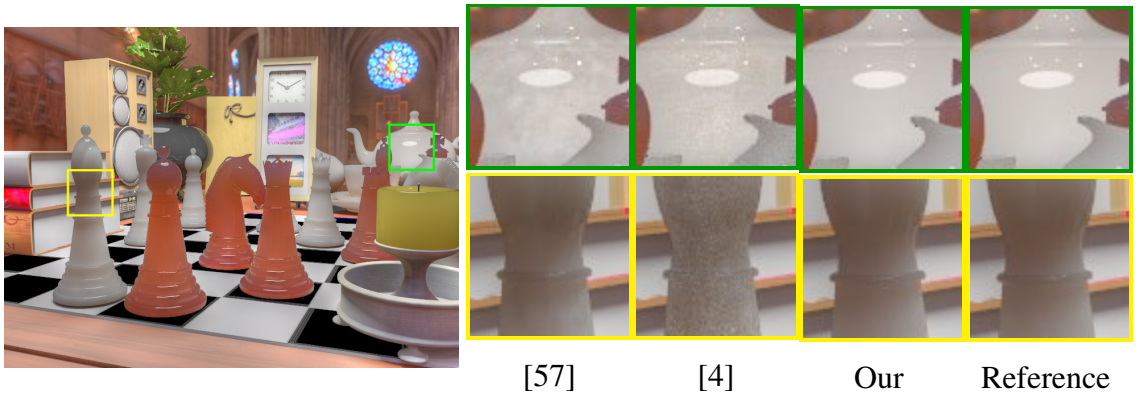


Figure 4.9: Equal-time comparison of the *ChessGame* scene (110 sec.). The RMSE values of images (a) to (c) are 0.003472, 0.005018, and 0.001196, respectively.

- Our method: our dual-matrix sampling method described in section 4.3. The number of light samples is set as the same as traditional two-pass method. For all scenes, we use 4,096 surface clusters and camera clusters. In our experiments, we found the number of clusters makes very slight difference to the final results. We also fix the number of sparse surface samples in irradiance estimation to 8, and the number of nearest neighbors K in diffuse BSSRDF estimation to 10.

Figure 4.6 to 4.10 show the equal-time comparisons of the five scenes of the three methods abovementioned. In this comparison,we first measured the time spent by Jensen and Buhler's method [57]. The refining threshold in adaptive diffusion approximation (Equation 4.5) is set to 0.05 (default value used in PBRT's implementation). The number

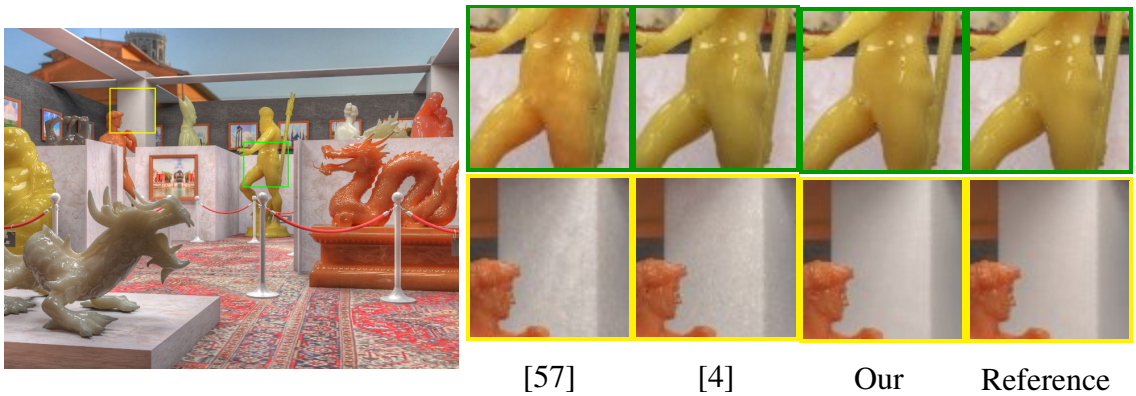


Figure 4.10: Equal-time comparison of the *Exhibition* scene (300 sec.). The RMSE values of images (a) to (c) are 0.011949, 0.016933, and 0.007185, respectively. Our method reduces the color shifts due to inaccurate surface irradiance.

of light samples in irradiance computation is set as follows: 50 for *Sculpture* and *Cathedral* scene, 200 for *ChessGame* scene, and 400 for *Room* and *Exhibition* scene. Then we carefully adjust parameters for Arbree *et al.*'s approach [4] and our method to match these time budgets.

In the *Sculpture* scene (Figure 4.6), because only a small fraction of surface samples (about 10%) locate in the view frustum, the traditional two-pass approach [57] which computes surface irradiance in a brute-force manner is very ineffective. The single-pass method based on lightcuts [4] avoids this over-computation by evaluating surface irradiance on demands surrounding camera samples. Although their method does improve performance in this case, it is still not comparable to our method because the surface irradiance cannot be reused directly. Our method possesses the advantages of traditional two-pass method (reuse surface irradiance) and the single-pass method (reduce unnecessary irradiance computation, 88% in this case) and achieves superior performance.

The *Cathedral* (Figure 4.7) and the *Room* scene (Figure 4.8) demonstrate two cases in which lightcut-based methods do not work well. In the *Cathedral* scene, the jade lucy is illuminated from the backside and almost all surface samples viewed by the camera have no contributions. Because the error metric used by subsurface lightcuts [4] does not

include a visibility term, it assumes that contribution will mainly come from the surface samples viewed by the camera for their larger diffuse BSSRDF terms. Similar situation occurs in the *Room* scene, where the occluded illumination from the environment map makes the refinement of light-surface-camera triples ineffective. The RMSE values listed below the figures show Arbree et al. 's approach [4] produces much higher error than the others. Compared to Jensen and Buhler's method [57], in these two scenes our method saves 70% and 32% irradiance computation, respectively. The reduced computation allows our method to concentrate resource on important surface samples (i.e.gather more lights). As a result, our method can largely reduce the color shifts due to inaccurate surface irradiance values produced by their method. This phenomena is especially obvious in scenes with complex lighting, such as the *Room* (Figure 4.8) and *Exhibition* (Figure 4.10) scene. Moreover, our error metric for adaptive diffusion approximation is also more accurate and improves the overall performance.

In the *ChessGame* scene (Figure 4.9), although the several translucent objects form a relatively complex layout, the occlusion is not severe and almost all translucent surfaces locate in the view frustum. In this case, subsurface lightcuts [5] and our method could only save about 45% irradiance computation. However, our error metric for adaptive diffusion approximation is much efficient than Jensen and Buhler's method [57] because it can better capture the high-frequncy changes in the illumination function.

Finally, in the *Exhibition* scene (Figure 4.10), because a lot of surface samples have little contributions to the final image due to occlusion, our method achieves great performance improvement compared to Jensen and Buhler's method [57]. In this scene, we save 70% irradiance computation. It is worth noting that although Arbree *et al.*'s method [4] also saves irradiance computation, the benefit was overwhelmed by the overhead of tree refinement and low hit-rate of form factor cache.

We also conduct equal-quality comparisons of the three methods. Figure 4.11 shows

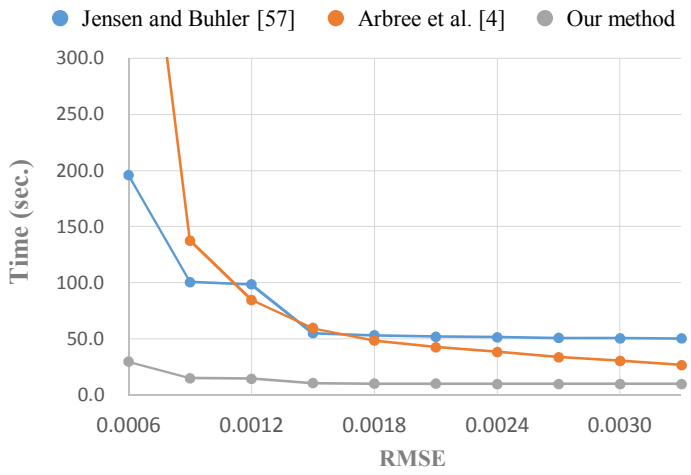
the time-RMSE plots of the Sculpture, Cathedral, and Exhibition scene. For two-pass methods (including Jensen and Buhler's approach [57] and our method), there are two ways to reduce error: increase light samples or lower the threshold for adaptive diffusion approximation. In this experiment, we start the assessment with a small number of light samples (50 for *Sculpture* and *Cathedral*, and 100 for *ChessGame*, *Room*, and *Exhibition* scenes). At a fixed number of light samples, we gradually lower the threshold of adaptive error approximation. The error decreases quickly at the beginning but will converge to one value due to inaccuracy of surface irradiance. When the error does not decrease obviously, we increase the number of light samples. This is the reason that why the required time jumps at some specific error. As the figures reveal, in most cases, two-pass methods are more efficient for reusing surface irradiance. Our method outperforms the other approaches consistently in all five scenes. It takes much less time to achieve a specific error.

4.5 Conclusion and Future Work

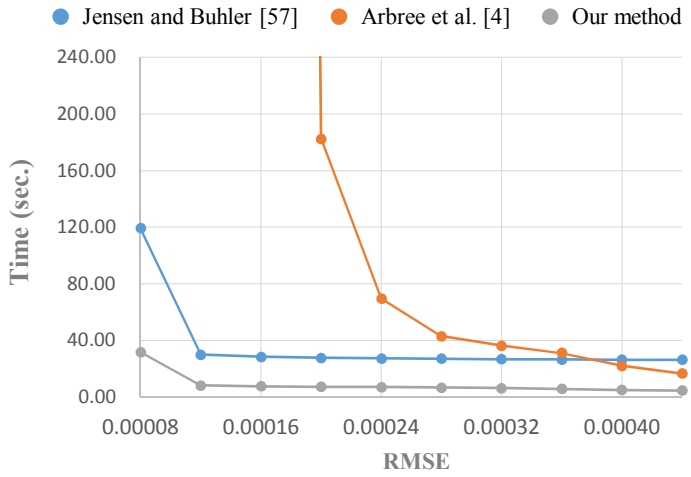
In this chapter we introduce a scalable algorithm for rendering translucent materials. Our method is based on the dual-matrix representation, which represents the light transport by two matrices: Light-to-Surface and Surface-to-Camera. The two matrices have distinct structures because of different characteristics of energy transport. We exploit such structures to avoid computing unnecessary surface irradiance that will not be used in the adaptive diffusion approximation. Our method is substantially faster because it not only reduces irradiance pre-computation but also provides a more accurate error estimation for the adaptive gathering.

In the future, we would like to extend our method to deal with the heterogeneous translucent materials. In this case, the bound of diffuse BSSRDF cannot be easily determined by the shortest distance of a camera sample to a surface cluster. We would

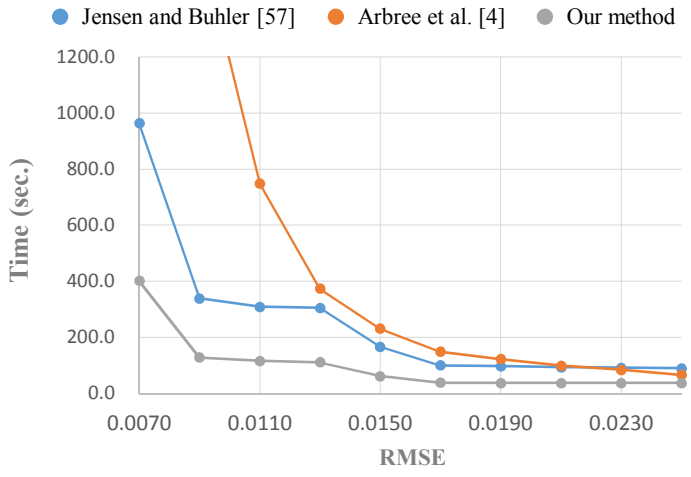
like to explore approaches for addressing this problem. It is also interesting to combine our method with the image-space caching approach [61], or the local BSSRDF sampling approaches [62].



(a) *Sculpture*



(b) *Cathedral*



(c) *Exhibition*

Figure 4.11: Equal-quality comparisons of Jensen and Buhler [57], Arbree et al. [4], and our method.

Chapter 5

SURE-based Optimization for Adaptive Sampling and Reconstruction

When several distributed effects including depth of field, motion blur, and global illumination should be simulated simultaneously, Monte Carlo integration is the most popular rendering approach for its generality and robustness. However, rendering a complex scene with multiple distributed effects usually requires several thousand expensive samples per pixel to produce a visually pleasing image. In this chapter, we introduce a novel rendering framework based on adaptive sampling and reconstruction to accelerate Monte Carlo rendering. Our method applies the Stein's Unbiased Risk Estimator (SURE), a general unbiased estimator for mean squared error (MSE) in statistics, to the rendering community. With SURE, we are able to estimate error for an arbitrary reconstruction kernel, enabling us to use more effective kernels rather than being restricted to the symmetric ones used in previous work. It also allows us to allocate more samples to areas with higher estimated MSE. We also propose an efficient and memory-friendly approach to reduce the impact of noisy geometry features where there is depth of field or motion blur. Experiments show that our method produces images with less noise and crisper details than previous methods.

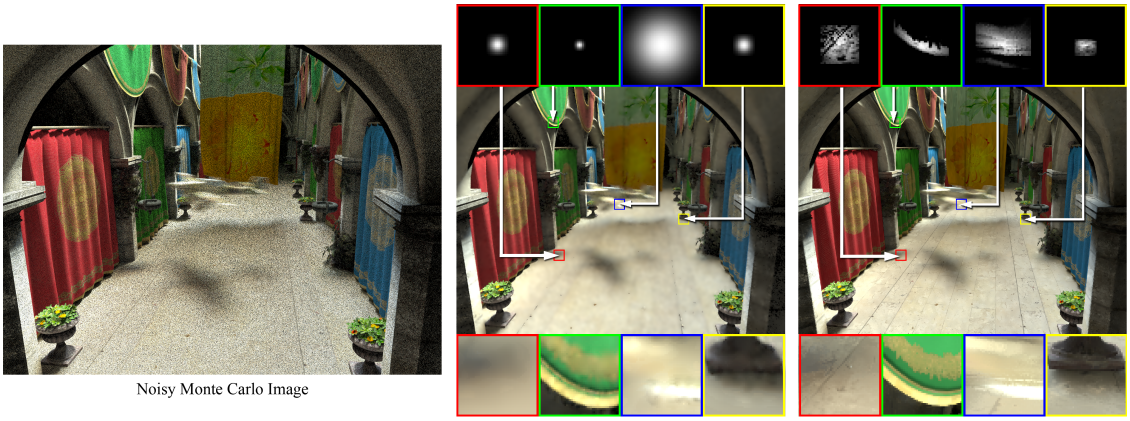


Figure 5.1: Comparisons between greedy error minimization (GEM) [87] and our SURE-based filtering. With SURE, we are able to use kernels (cross bilateral filters in this case) that are more effective than GEM's isotropic Gassians. Thus, our approach better adapts to anisotropic features (such as the motion blur pattern due to the motion of the airplane) and preserves scene details (such as the textures on the floor and curtains). The kernels of both methods are visualized for comparison.

5.1 Overview

As discussed previously, Monte Carlo (MC) integration is a common technique for rendering images with distributed effects such as antialiasing, depth of field, motion blur, and global illumination. It simulates a variety of sophisticated light transport paths in a unified manner; it estimates pixel values by using stochastic point samples in the integral domain. Despite its generality and simplicity, however, the MC approach converges slowly. A complex scene with multiple distributed effects usually requires several thousand expensive samples per pixel to produce a noise-free image.

Adaptive sampling and reconstruction (or filtering, used interchangeably in the chapter) are two effective techniques for reducing noise. Given a fixed budget of samples, adaptive sampling determines the optimal sample distribution by concentrating more samples on difficult regions. To decide which pixels are worth more effort, we require a robust criterion for measuring errors. Accurate estimation of errors is challenging in our appli-

cation because the ground truth is not available. Reconstruction algorithms, in contrast, properly construct smooth results from the discrete samples at hand. One key issue that reconstruction must resolve is how to select the filters for each pixel, as the optimal reconstruction kernels are usually spatially-varying and anisotropic. Recently, approaches have been developed to address the challenge of spatially-varying filters [16, 87], producing better results than those that use a single filter across the whole image. However, these methods are limited to symmetric filters and do not work well for scenes with anisotropic features such as high-frequency textures on the floor and curtains in Figure 5.1.

We here propose an adaptive sampling and reconstruction algorithm to improve the efficiency of Monte Carlo ray tracing. The core idea is to adopt Stein's Unbiased Risk Estimator (SURE) [97], a general unbiased estimator for mean squared error (MSE) in statistics, to determine the optimal sample density and per-pixel reconstruction kernels. The advantages of using SURE are twofold. For one, it provides a means by which to measure the quality of arbitrary reconstruction kernels – not just those that are symmetric (e.g. isotropic Gaussians used in greedy error minimization (GEM) [87]). As such, it allows for the use of more effective filters such as cross bilateral filters and cross non-local means filters. Another advantage is that the per-pixel errors estimated by SURE can be used to guide further sample distribution. Thus more samples can be allocated to difficult regions. In addition to applying SURE, we propose an efficient and memory-friendly approach to maintain the quality of cross bilateral filtering in the presence of noisy geometric features when rendering depth-of-field or motion blur effects. We propose a normalized distance to alleviate the impact of noisy features, making the proposed method more robust to all types of distributed effects. Experiments show the proposed method provides significant improvements over previous techniques for adaptive sampling and reconstruction.

The rest of this chapter is organized as follows: First, in Section 5.2, we review the most important previous work in the context of adaptive sampling and reconstruction. We

then introduce the theorem of SURE in Section 5.3. The algorithm and implementation details of the proposed rendering framework are given in Section 5.4. In Section 5.5, we provide the performance assessment of our method. Comparison between two the state-of-the-art methods and our method are also included. Finally, we conclude the contribution of our work in Section 5.6.

5.2 Related Work

5.2.1 Adaptive sampling and reconstruction

The seminal work of Mitchell [72, 73] over twenty years ago laid the foundation for methods for adaptive sampling and reconstruction. We categorize these techniques as image space methods, multidimensional methods, and adaptive filtering.

Image space methods estimate per-pixel errors with various criteria and allocate additional samples to difficult regions. These approaches usually are more general and can be used for various types of effects. Bala *et al.* [5] combine edge information and sparse point samples to perform edge-aware interpolation. The quality of their results is highly dependent on the accuracy of edge detection. Overbeck *et al.* [81] proposed adaptive wavelet rendering (AWR), a general wavelet-based adaptive sampling and reconstruction framework. They distinguish the sources of variances as the coarse level for distributed effects and the fine level for edges. These two types of noise are addressed separately by sampling the hierarchical wavelet coefficients adaptively; smooth images are reconstructed by thresholding wavelet coefficients. Recently, several approaches [16, 87] have been proposed to smooth out noise using multi-scale filters. Chen *et al.* [16] focus on depth of field and describe a criterion to select spatially-varying Gaussian filters from a predefined filterbank based on the depth map. Rousselle *et al.* [87] also use Gaussian filters to form a filterbank. Although their method uses an error minimization framework

for adaptive sampling and reconstruction, and is more general, the framework can be used only for symmetric filters. We apply SURE to estimate MSE for more general filters and allow the use of more effective filters, which yield significant improvements.

Other approaches perform adaptive sampling and reconstruction in a multidimensional space. Hachisuka *et al.* [50] distribute more samples to discontinuities in the high-dimensional space and reconstruct them anisotropically using structure tensors. Their approach achieves good quality but becomes less efficient as the number of dimensions increases. Approaches have also been developed that are based on transform domain analysis, focusing on specific distributed effects such as depth of field [96], motion blur [42], and soft shadows [41]. Recently Lehitinen *et al.* proposed a novel method for reconstructing temporal light-field samples [66], and later extended it to handle the indirect light field for global illumination [67]. By reprojecting samples along the sample trajectory in the multidimensional space, expensive samples can be reused. In general, to avoid the curse of dimensionality, multidimensional approaches usually focus on only one or two specific effects. These methods are able to generate better results for the effects they focus on because the anisotropy of the integrand is taken into account. Our method, however, is more general, efficient, and memory-friendly.

Some methods focus on reconstruction only. Xu *et al.* [111] proposed smoothing out Monte Carlo noise with a modified bilateral filter with smoothed range values. Focusing on interactive global illumination, Segovia *et al.* [88], Dammertz *et al.* [28], and Bauszat *et al.* [8] exploit geometric properties such as depths and surface normals to identify edges. Shirley *et al.* [92] use the depth buffer to help filtering, but target defocus and motion blur. Sen and Darabi [89] proposed a general adaptive filtering approach based on information theory. By identifying the dependencies between random parameters on sample colors and scene features, they reduce the importance of sample values influenced by Monte Carlo noise.

5.2.2 Denoising using SURE

Stein [97] proposed an MSE estimator called Stein's Unbiased Risk Estimator (SURE) for estimators on samples with normal distributions. Donoho and Johnstone [38] incorporate the estimator into a Wavelet shrinkage algorithm to reconstruct functions from noisy inputs. Recently, SURE has received much acclaim from the image denoising community and has been widely used to optimize denoising parameters [10, 103].

5.3 Stein's Unbiased Risk Estimator (SURE)

Monte Carlo ray tracing estimates true pixel colors by randomly sampling the integral domain and reconstructing from samples. The unbiased Monte Carlo rendering techniques have a stochastic error bound which can be estimated using the variance of the estimator. According to the central limit theorem, if Y is the pixel color estimated by an unbiased Monte Carlo renderer and x is the true color, as the number of samples n approaches infinity, Y's distribution approximates a normal distribution with mean x and variance σ^2/n:

$$Y \xrightarrow{d} N\left(x, \frac{\sigma^2}{n}\right), \tag{5.1}$$

where σ^2 is the variance of the Monte Carlo samples. Previous investigations have demonstrated that, for a finite number of samples, this relationship is still a good approximation [101, 44, 49].

Since a Monte Carlo renderer is an estimator, it is often necessary to estimate its accuracy for applications such as adaptive sampling. Stein's Unbiased Risk Estimator (SURE) offers a means for estimating the accuracy of a given estimator [97, 10]. SURE states that, if y is a measurement on x with a normal distribution $N(x, \sigma_y^2)$, and F is a weakly differ-

entiable function, then the following estimation of error[1],

$$\text{SURE}(F(y)) = \|F(y) - y\|^2 + 2\sigma_y^2 \frac{dF(y)}{dy} - \sigma_y^2, \tag{5.2}$$

is an unbiased estimator of the mean square error (MSE) of F(y), that is,

$$E[\text{SURE}(F(y))] = \|F(y) - x\|^2. \tag{5.3}$$

Equation 5.2 and 5.3 indicate that if we can compute σ_y and $dF(y)/dy$, we can estimate the error of an estimator F without knowing the true underlying value x.

Our goal is to use the above formula to estimate the reconstruction error of an arbitrary kernel F. As mentioned above, rendering samples y follow a normal distribution. Thus, SURE can be used as the estimator for MSE of any filter F if we can compute $dF(y)/dy$ for the reconstruction kernel (σ_y can be directly estimated from Monte Carlo samples). Section 5.4 describes details of the algorithm, including the definition of F, the derivation of $dF(y)/dy$, and how to use SURE for optimal filter selection and adaptive sampling.

It is worth noting that Rousselle *et al.* [87] attempted to estimate MSE for the same application. They decomposed the error into variance and bias and exploited the relation between the biases of the two filters to estimate the error. However, they used a quadratic approximation which is valid only for symmetric filter kernels, thus limiting the effectiveness of their approach. In contrast, our SURE-based estimation works for arbitrary reconstruction kernels and provides more flexibility to the choice of kernels.

5.4 SURE-based Adaptive Rendering

Figure 5.2 demonstrates the flowchart of the proposed method. Our method starts by rendering a small number of initial samples for each pixel and then iterates between

[1]We followed Blu and Luisier's SURE formulation [10], which was derived from the original SURE [97]; their equivalence has been shown. Since we apply SURE to estimate MSE errors for each color channel independently, the filter kernel F is a scalar function. Thus, the dimension is 1 and the divergence in the original SURE formulation becomes a derivative.

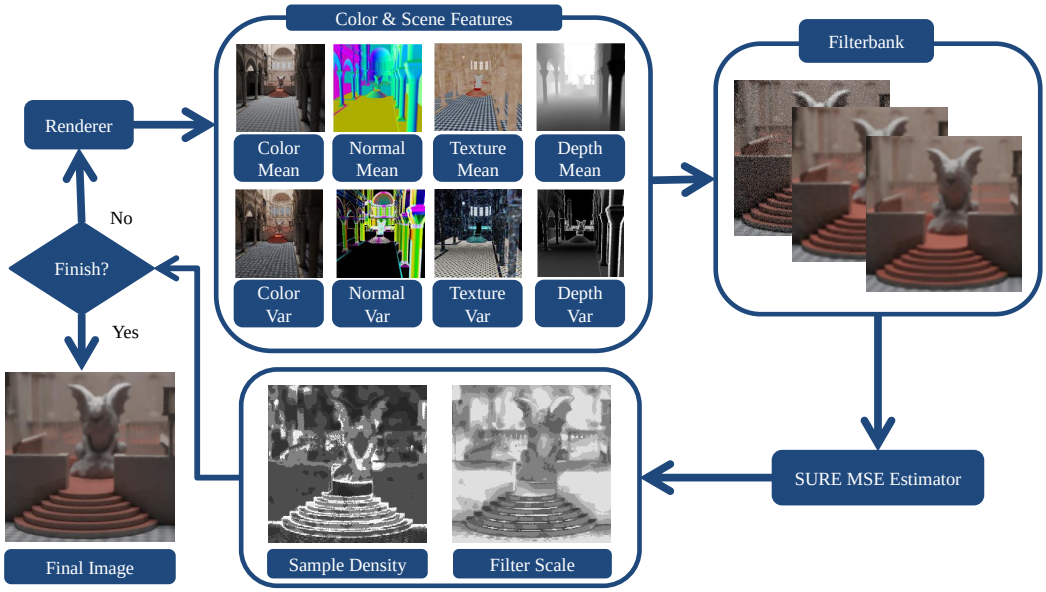


Figure 5.2: An overview of our algorithm, which alternates between sampling and reconstruction until reaching the sample budget. During sampling, a set of samples collects colors, normals, textures and depths over the image plane. They are used as the side information for filters in the reconstruction stage, during which, for each pixel, a set of filters are performed and the filtered value with the minimal SURE value is filled into the reconstructed image. In addition, the minimal SURE value of each pixel is recorded to guide adaptive sampling. If there are sample budgets left, more samples are shot for pixels with larger SURE errors.

filter selection and adaptive sampling stages until reaching the sample budget. After each sampling phase, we reconstruct a set of images using a filterbank. Each pixel is filtered multiple times with all candidate filters in the filterbank and the error of each filtered pixel color is estimated by computing SURE. For each pixel, the filtered color with the least SURE error is chosen and filled into the reconstructed image. If more sample budget is available, the adaptive sampling stage is invoked and a batch of new samples is distributed

to pixels with larger SURE errors. Details are in the following subsections.

5.4.1 Initial samples

At the beginning, a small number of initial samples (usually 8 or 16 samples per pixel) are taken to explore the scene. The samples are generated by low discrepancy sequences and distributed evenly to each pixel. After rendering with these samples, our method performs the first reconstruction phase with the gathered information.

5.4.2 Filter selection using SURE

The main advantage of our method over greedy error minimization [87] is its ability to use arbitrary filters. We have experimented with three different filters: isotropic Gaussian, cross bilateral, and a modified non-local means filter [12] with additional scene feature information (we call this a "cross non-local means filter"; see Section 5.5.4). Here, we use the cross bilateral filter as an example since it offers the best compromise between performance and quality amongst these filters. Algorithms for different filters are the same except that different filters are used for constructing the filterbank; the derivatives in SURE are different also.

Cross bilateral filters

Cross bilateral filters have been shown effective for removing Monte Carlo noise [28, 89]. Similar to previous work, auxiliary data including surface normals, depths, and texture colors are collected by caching information after tracing rays. We compute the per-pixel mean and variance of each feature and store them as feature vectors for cross bilateral filters. For the cross bilateral filter, the filter weight w_{ij} between a pixel i and its neighbor j is defined as

$$\exp\left(-\frac{\|p_i - p_j\|^2}{2{\sigma_s}^2}\right)\exp\left(-\frac{\|c_i - c_j\|^2}{2{\sigma_r}^2}\right)\prod_{k=1}^{m}\exp\left(-\frac{D(\bar{f}_{ik},\bar{f}_{jk})^2}{2{\sigma_{f_k}}^2}\right), \tag{5.4}$$

where $\bar{f}_{ik}$ is the sample mean of the k-th feature for the pixel i; σ_s, σ_r, and σ_{f_k} are the standard deviation parameters of the spatial, range (sample color), and feature terms respectively. D is a distance function used to address noisy scene features for depth of field and motion blur. It will be discussed later in this section. The filtered pixel color $\hat{c}_i$ of the pixel i is computed as the weighted combination of the colors c_j of all neighboring pixels j:

$$\hat{c}_i = \frac{\sum_{j=1}^{n} w_{ij} c_j}{\sum_{j=1}^{n} w_{ij}}. \tag{5.5}$$

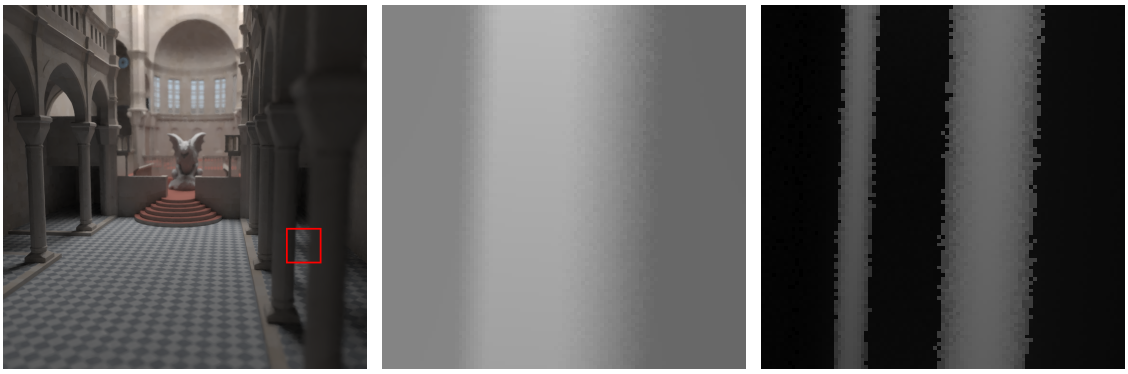
Note that the cross bilateral kernels are spatially-varying due to the range and feature terms. Figure 5.1 shows examples.

In our current implementation, the filterbank is composed of cross bilateral filters with different σ_s, corresponding to different spatial scales. Other parameters σ_r and σ_{f_k} are fixed and their values are discussed in Section 5.5.

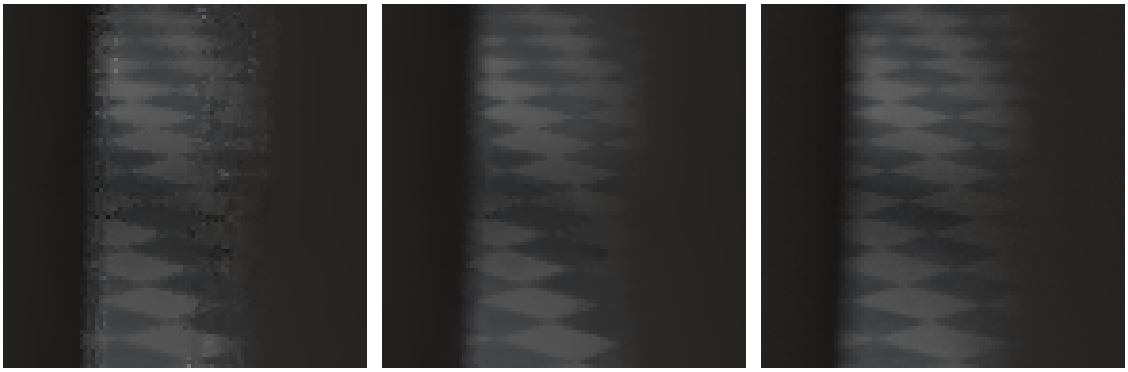
Depth of field and motion blur

Sen and Darabi [89] point out that when rendering depth of field and motion blur effects, the geometric features (surface normal and depth) can be noisy due to Monte Carlo noise. In these situations, as the weighting function is not accurate, using the features dogmatically for filtering can fail to remove the noise. They resolve this problem by computing the functional dependency of the Monte Carlo random parameters and the scene features and using it to reduce the weight of samples if their features are highly dependent on the random parameters. Although their method successfully handles depth of field and motion blur, it operates at the sample level. Thus, performance and memory consumption become issues since computing pairwise mutual information between samples and parameters is not only time-consuming but also requires considerable memory for storing samples.

We propose a more efficient and memory-friendly approach to prevent cross bilateral



(a) *Sibenik* (b) Depth (c) Depth variance



(d) L2 distance (e) Normalized distance (f) Reference

Figure 5.3: Comparisons of filtering the *Sibenik* scene with depth-of-field effects using L2 distance and our normalized distance. The area with strong depth of field has noisy geometry information, preventing us from filtering when using L2 distance between sample means. By incorporating sample variances, normalized distance allows us to filter these areas even given noisy geometry information. Note that the large depth variances in (c) allow us to filter areas around the pillars, removing the artifacts exhibited in the result with L2 distance (d).

filters from being affected by noisy scene features. Each pixel has a set of samples for feature k. Given two pixels i and j, to measure their distance with regard to the feature, the naive metric would be the distance between the sample means, $\bar{f}_{ik}$ and $\bar{f}_{jk}$. This, however, completely ignores sample variances. If we model samples as Gaussians, the distance between two sample sets should be normalized by their variances. Thus we define the

normalized distance as

$$D(\bar{f}_{ik}, \bar{f}_{jk}) = \sqrt{\frac{\|\bar{f}_{ik} - \bar{f}_{jk}\|^2}{\sigma_{ik}^2 + \sigma_{jk}^2}}, \tag{5.6}$$

where σ_{ik}^2 and σ_{jk}^2 are sample variances of the k-th feature of pixels i and j respectively. Intuitively, for a pixel with strong depth of field and motion blur, its samples tends to have a large variance since these samples usually span over a large region in the spatial-temporal domain. Thus, it tends to have smaller distances and larger weights even when the geometric features are far apart. In the extreme case that two feature sets are inseparable due to strong depth of field or fast motion, the cross bilateral filter reduces to a Gaussian filter and does not use the unreliable geometric features. This approach allows us to evaluate feature importance at the pixel level and store only the sample mean and variance of features per pixel. Figure 5.3 shows the effect of the proposed distance metric.

Computing SURE and selecting the per-pixel optimal filter

For each pixel, we need to use the minimal SURE error to determine the optimal scale for the cross bilateral filters in the filterbank. As mentioned in Section 5.3, in calculating SURE, we need to compute $dF(c_i)/dc_i$ for the cross bilateral filter F defined in Equation 5.5. We have obtained its analytic form as

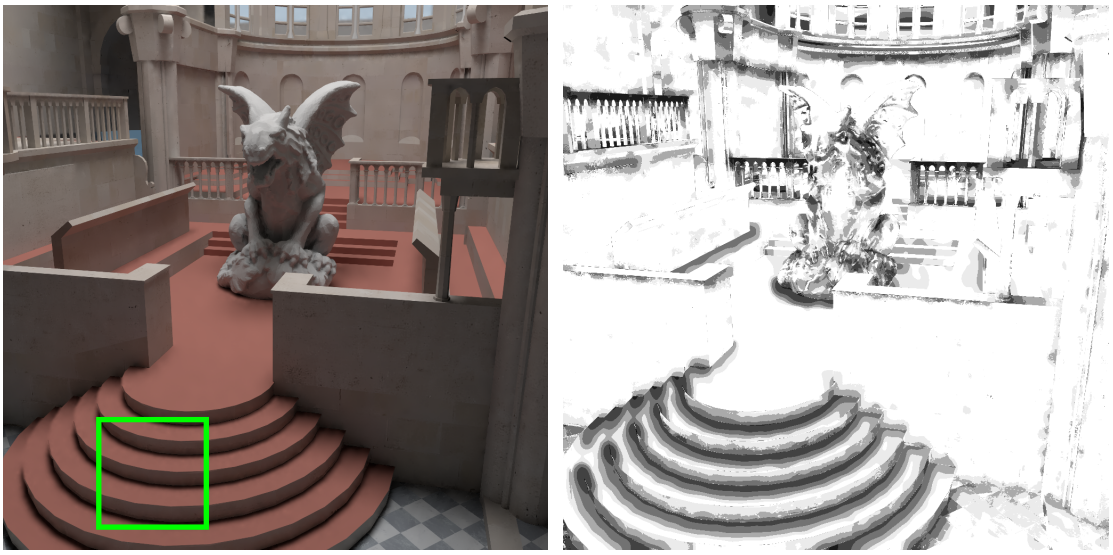
$$\frac{dF(c_i)}{dc_i} = \frac{1}{\sum_{j=1}^{n} w_{ij}} + \frac{1}{\sigma_r^2}(F^2(c_i) - F(c_i)^2), \tag{5.7}$$

where

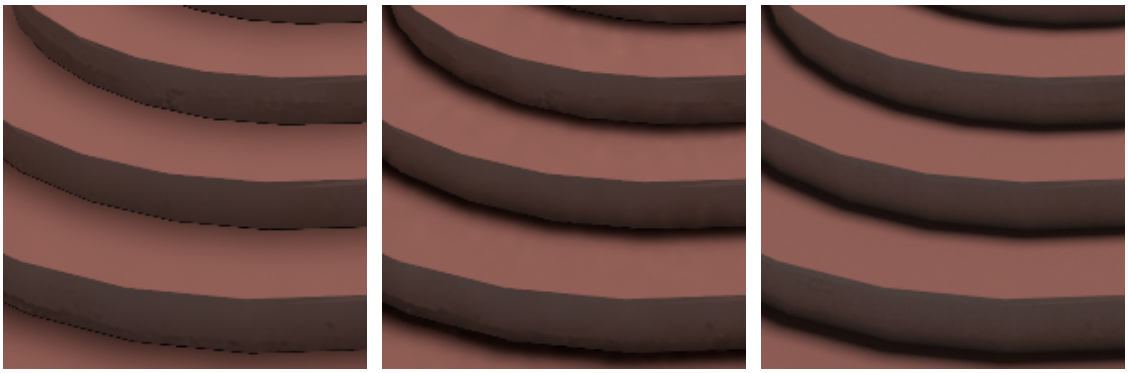
$$F^2(c_i) = \frac{\sum_{j=1}^{n} w_{ij} {c_j}^2}{\sum_{j=1}^{n} w_{ij}}. \tag{5.8}$$

The derivation is in Appendix A. We then compute SURE to estimate the MSE for each filter in the filterbank using Equation 5.2. For each pixel, the filter with the least SURE error is selected and its filtered color is used to update the pixel.

We have observed that computing SURE using Monte Carlo samples usually leads to noisy filter selection and thus yields noisy results. This is because SURE is an unbiased



(a) *Gargoyle* (b) Scale selection map



(c) Global, MSE: 0.011718 (d) Our, MSE: 0.002148 (e) Reference

Figure 5.4: Visualization of the scale selection map for σ_s of our method. We have also compared our approach to a global cross bilateral filter (which uses the same scale parameter, the largest σ_s in the filterbank, for all pixels). It is clear that the global cross bilateral filter produces large bias in the shadow areas. Our approach adapts better in these areas and leads to a smaller error.

estimator of MSE and has its own variances. To reduce variances, one can either add more samples or perform filtering. For the sake of efficiency, we opted to perform filtering to reduce the variances of SURE. To be more concrete, we prefilter the estimated MSE image using a cross bilateral filter with a fixed parameter before SURE optimization. A similar problem was encountered in the previous method [87]; they smoothed out the

selected scales of filters to deal with the variance of their estimator.

Figure 5.4 shows the scale selection map and compares our SURE-based filtering with a global cross bilateral filter (with the same scale for each pixel). It is obvious that spatially-varying scale selection yields better results both visually and quantitatively.

5.4.3 Adaptive sampling

The MSE estimated using SURE can be taken as feedback to the renderer; the sampling density should be proportional to the estimated MSE. However, since our MSE estimation is not perfect (note that Equation 5.2 can be negative), a heuristic variance term is included to ensure that regions with higher variances are allocated more samples. In addition, to guide more samples to darker areas, we scale our sampling function with the squared luminance of the filtered color. This strategy was also adopted by a previous approach [81] because human eyes are more sensitive to error in dark regions. As a result, the sampling function for a pixel i is determined by

$$S(i) = \frac{\mathrm{SURE}(F(c_i)) + \sigma_i^2}{I(F(c_i))^2 + \varepsilon}, \tag{5.9}$$

where σ_i^2 is the variance of samples within the pixel, $I(F(c_i))^2$ is the squared luminance of the filtered pixel color, and ε is a small number used to prevent a null denominator (set to 0.001). If the current sampling budget is m, pixel i receives $\lceil mS(i)/\sum_j S(j) \rceil$ samples. Figure 5.5 visualizes the sampling density for two examples. It is clear that samples concentrate on areas with geometry or texture details, discontinuities, or more noise.

5.5 Experiments

We implemented the algorithm on top of the PBRT2 system [82]. All results were generated on a machine with an Intel dual quad-core Xeon E5420 CPU at 2.5GHz, 32GB of RAM, and using 8 threads. As mentioned, we mainly used cross bilateral filters in the

Sampling density Reconstructed image

Figure 5.5: Visualizations for the sampling density of our approach.

proposed SURE-based framework. In Section 5.5.4 we discuss results with other filters.

5.5.1 Parameter setting

There are a number of parameters for the features in Equation 5.4. They were set as $\sigma_{fk} = 0.8$ for normal, $\sigma_{fk} = 0.25$ for texture color, and $\sigma_{fk} = 0.6$ for depth throughout all experiments. We did not use σ_r in our current implementation since in practice we found the color term in the cross bilateral filter does not help much. We varied the spatial scale parameter σ_s to form the filterbank. We used $\sigma_s = 1, 2, 4$ to construct the filterbank in intermediate iterations and $\sigma_s = 1, \sqrt{2}, 2, 2\sqrt{2}, 4, 4\sqrt{2}, 8$ for the final reconstruction. We used fewer filters for intermediate phases as we found it sufficient and more efficient. Experi-

ments show this setting strikes a good compromise between performance and quality. For the parameters used in prefiltering before SURE computation, we set $\sigma_s = 8$, and the same σ_{fk} as mentioned above. In practice, results are not very sensitive to these parameters and a wide range of parameters work equally well. Although parameters can be fine-tuned for each scene, this yields only marginal improvements.

5.5.2 Comparisons

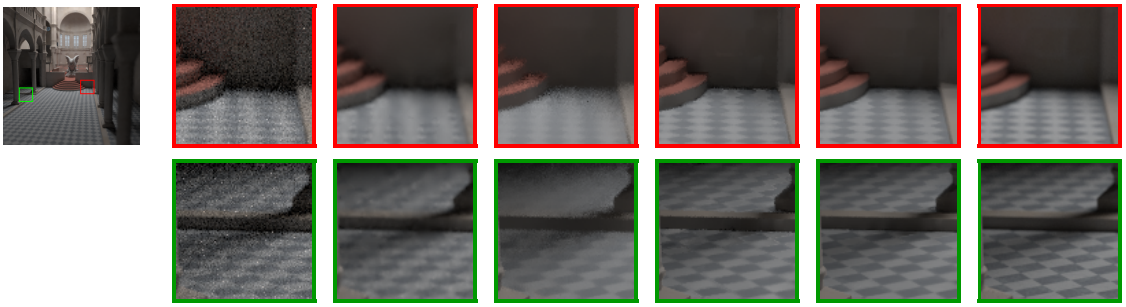
We applied our algorithm on rendering four scenes – *Sibenik* (1024x1024), *Teapot* (800x800), *Sponza* (1600x1200) and *Town* (800x600) – with a variety of effects, including global illumination, motion blur, depth of field, area lighting, and glossy reflection (Figure 5.6 to 5.9). We have also compared our method on these scenes with the following methods:

- MC: Uniform sample distribution and per-pixel box filter. This approach is used as the baseline without adaptive sampling and reconstruction.

- GEM: Adaptive sampling and reconstruction using greedy error minimization [87]. The results were produced by the authors' implementation on the PBRT2 system. For all scenes, we set the γ parameter in their algorithm to 0.2 as the paper suggested.

- RPF: Adaptive filtering using random parameter filtering [89]. We implemented their approach on the PBRT2 system. The σ^2 in their algorithm is set to 0.002 according to the authors' suggestion.

The number of samples for each method was carefully adjusted to make equal-time comparisons. However, since RPF consumes considerable memory and time compared to other methods, its number of samples was limited to 8 or 16. For very complex scenes, the time for reconstruction could be negligible if taking samples is very expensive. To

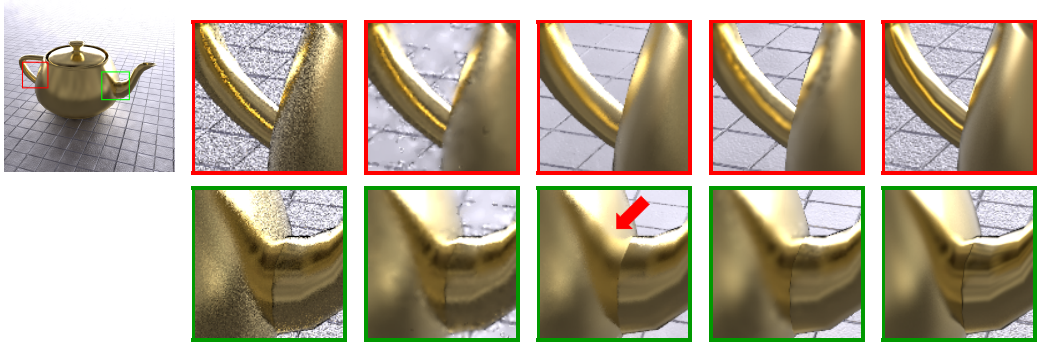


Sibenik	44 spp	39.86 spp	8 spp	8 spp	26.69 spp	4096spp
time	140s	135s	363s	64.2s	140s	
relative MSE	0.029946	0.002070	0.006103	0.003100	0.001489	

Figure 5.6: A comparison on the *Sibenik* scene with global illumination and depth of field. GEM adapts poorly to the texture on floor and produces oversmoothed results. RPF detects high dependency between u-v parameters and the color, thus filtering the area heavily and also producing oversmoothed results. The RPF image noise is from the sampling approximation of the bilateral filter.

make fair comparisons under such situations, we also include equal-sample comparisons between RPF and our method. Finally, we also compared all methods quantitatively with the relative MSE proposed by Rousselle *et al.* [87]. It is defined as the average of $(y - x)^2/(x^2 + \varepsilon)$, where y is the estimated pixel color, x is the pixel color in the reference image, and ε is set to 0.01 to prevent division by zero.

Figure 5.6 compares these algorithms on *Sibenik*, a scene with global illumination and depth of field. The image produced by MC retains considerable high-frequency noise even in simple areas such as the floor. GEM eliminates floor noise, while at the same time oversmoothing the area with textures due to its use of isotropic filters. Note that, although its relative MSE seems good, GEM tends to yield oversmoothed images. RPF produces a slightly sharper image than GEM but it is still oversmoothed, especially where there are depth-of-field effects. Our approach produces an image with much less noise while faithfully preserving textures.



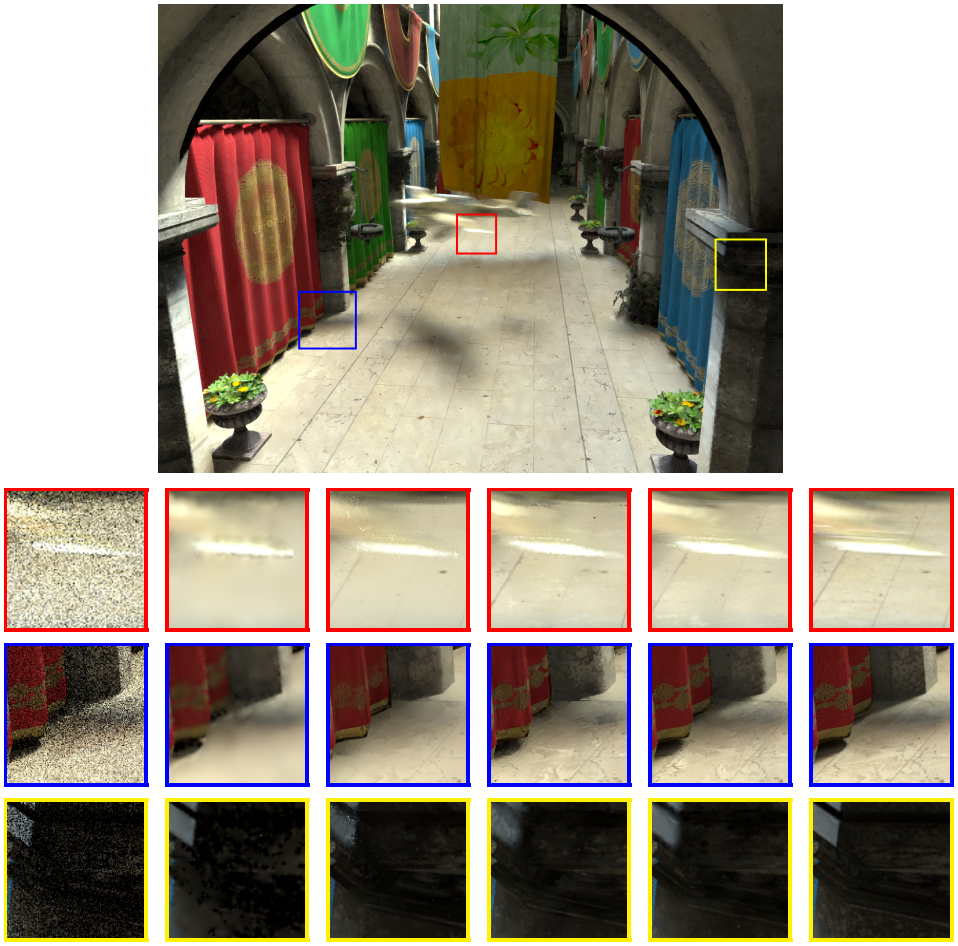
Teapot	35 spp	23.96 spp	8 spp	8 spp	4096spp
time	42s	44.3s	374.4s	40.4s	
relative MSE	0.199485	0.171002	0.233701	0.143123	

Figure 5.7: A scene with a glossy teapot. The floor contains complex texture and bump maps. All methods oversmooth the floor. RPF also oversmooths the glossy self-reflection of the teapot indicated by the arrow.

The *Teapot* scene (Figure 5.7) demonstrates a challenging case with very high-frequency bump mapping and glossy reflections. None of the four methods preserve the bump map on the floor well. Again, MC produces a very noisy image. It is also worth noting that RPF fails to reproduce the self-reflection on the teapot. Overall, our approach still produces an image that is visually more pleasing and quantitatively more accurate than other methods.

The *Sponza* scene in Figure 5.8 contains motion blur effects. As shown in the first row of insets, the anisotropic pattern produced by motion of the wing is more vivid in our result than in the others. In addition, our approach more faithfully preserves the textures on the floor and the curtains.

Finally the *Town* scene shown in Figure 5.9 was designed to test environment lighting, area lights, and motion blur. The scene is challenging also due to the heavy occlusion between the buildings and skyscrapers. Despite its strong MSE, GEM fails to reconstruct all the textures in the scene, which are preserved well in our results. RPF, on the other

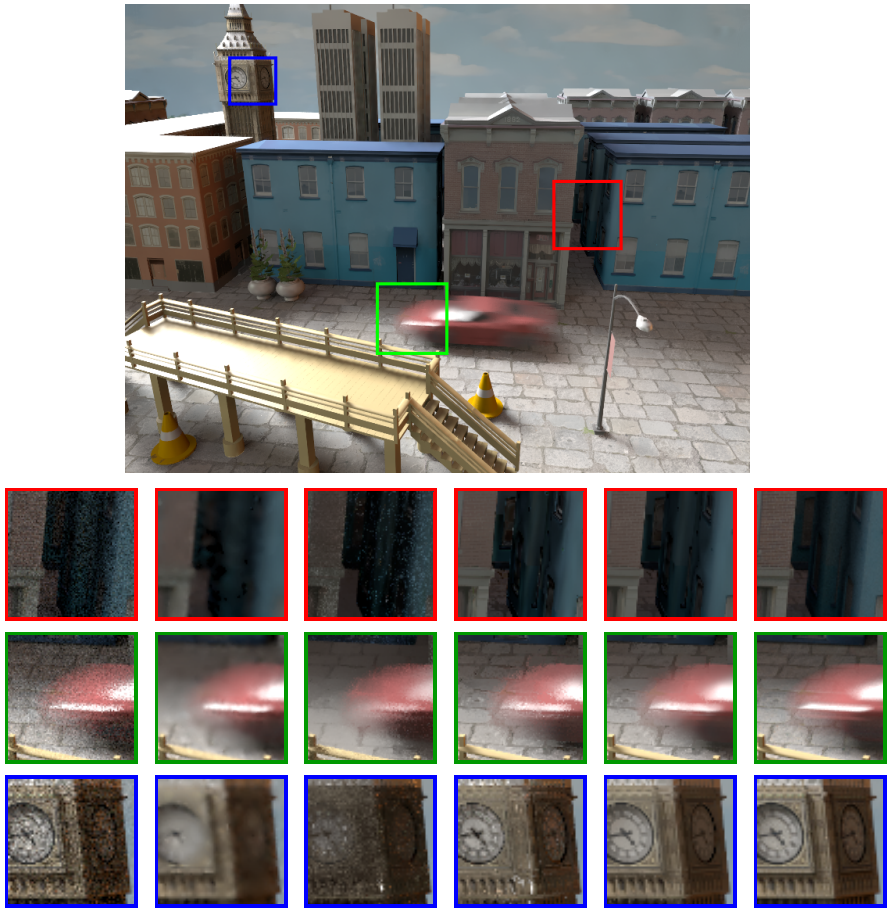


MC	GEM	RPF	Our	Our	Reference
68.0spp	63.8spp	16.0spp	16.0spp	63.2spp	8192.0spp
890.5s	906.2s	1676.1s	273.3s	896.0s	
0.133096	0.017605	0.031972	0.020549	0.012097	

Figure 5.8: Comparisons on a complex scene *Sponza* with global illumination and motion blur. The image on the top is our result. Insets show that GEM does not preserve details with symmetric filters, while RPF tends to oversmooth the shadows.

hand, produces a very noisy image. This could be related to the sampling procedure in their bilateral filtering computation. Our approach outperforms the others by producing less noise and crisper details.

In addition to the comparison on visual quality listed above, we also visualize the error map of each method in Figure 5.10.



MC	GEM	RPF	Our	Our	Reference
82 spp	51.82 spp	8 spp	8spp	39.79 spp	4096 spp
59.9s	61.8s	272.4s	20.0s	60.9s	
0.029797	0.018352	0.057937	0.034708	0.018023	

Figure 5.9: Comparisons on the *Town* scene with an environment light, an area light, and heavy occlusion. GEM fails to adapt to textures, and RPF does not obtain enough samples to reconstruct the scene within the given time.

5.5.3 Discussions

GEM performs adaptive sampling and selects per-pixel filters in an attempt to minimize MSE. From the results, it does achieve lower relative errors compared to MC and RPF (and comparable to our approach). However, as mentioned, GEM is limited to symmetric filters and does not adapt well to high-frequency textures and detailed scene features. In all our test scenes, the results produced by GEM exhibit obviously oversmoothed

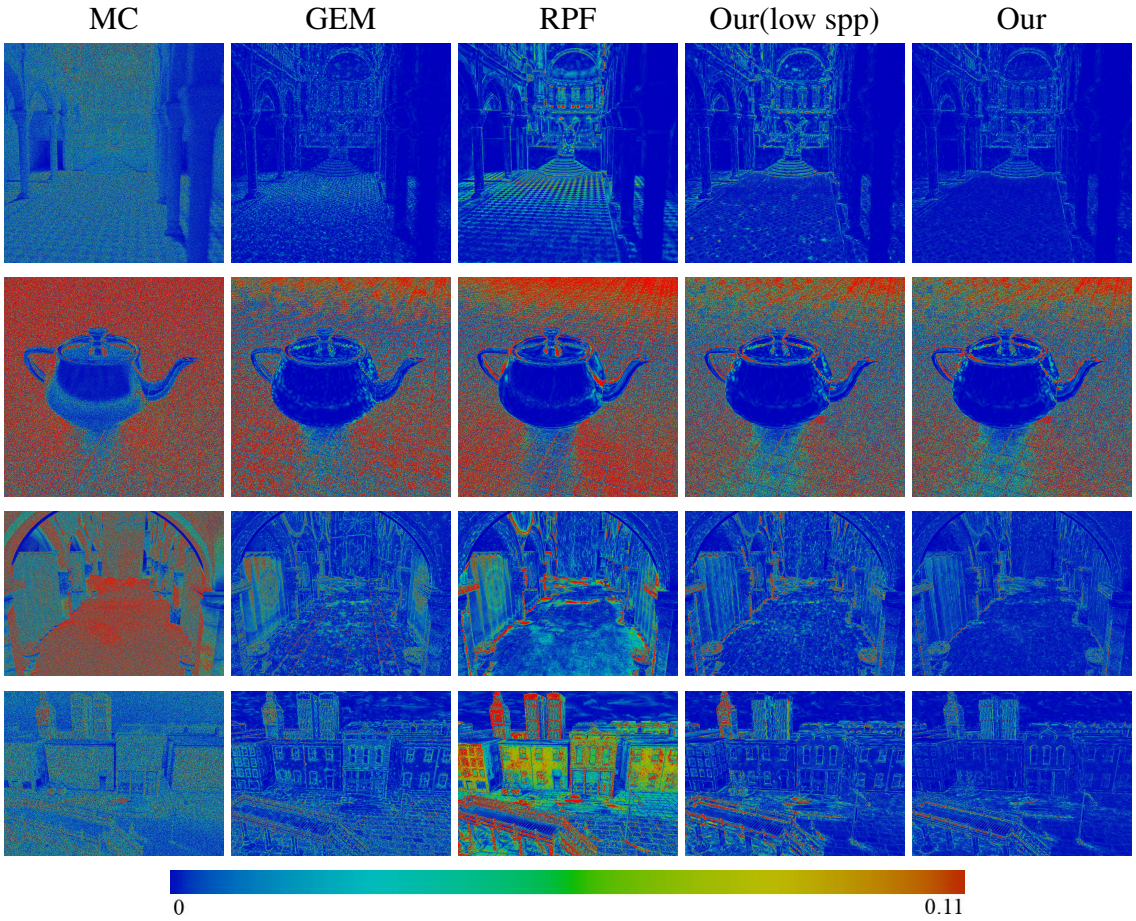


Figure 5.10: This figure visualizes the per-pixel relative error of each method. Scenes from the top: *Sibenik*, *Teapot*, *Sponza*, *Town*. The images show that overall our method produces lower error compare to other methods. It is worth to note that GEM gives lower error in a few regions, such as the shadowed area of the elevated walkway in the *Town* scene, this is because GEM can have more samples under an equal-time comparison, and our feature buffer does not capture the edge well in these region. Still, our method produces a much higher quality image overall compare to GEM.

artifacts. In addition, the GEM adaptive sampling criterion tends to send very few rays to the regions where most of the samples carry null radiance (for example, the right pillar of *Sponza* in Figure 5.8). Our approach significantly alleviates these problems by using cross-bilateral filters and prefiltering MSE before SURE optimization.

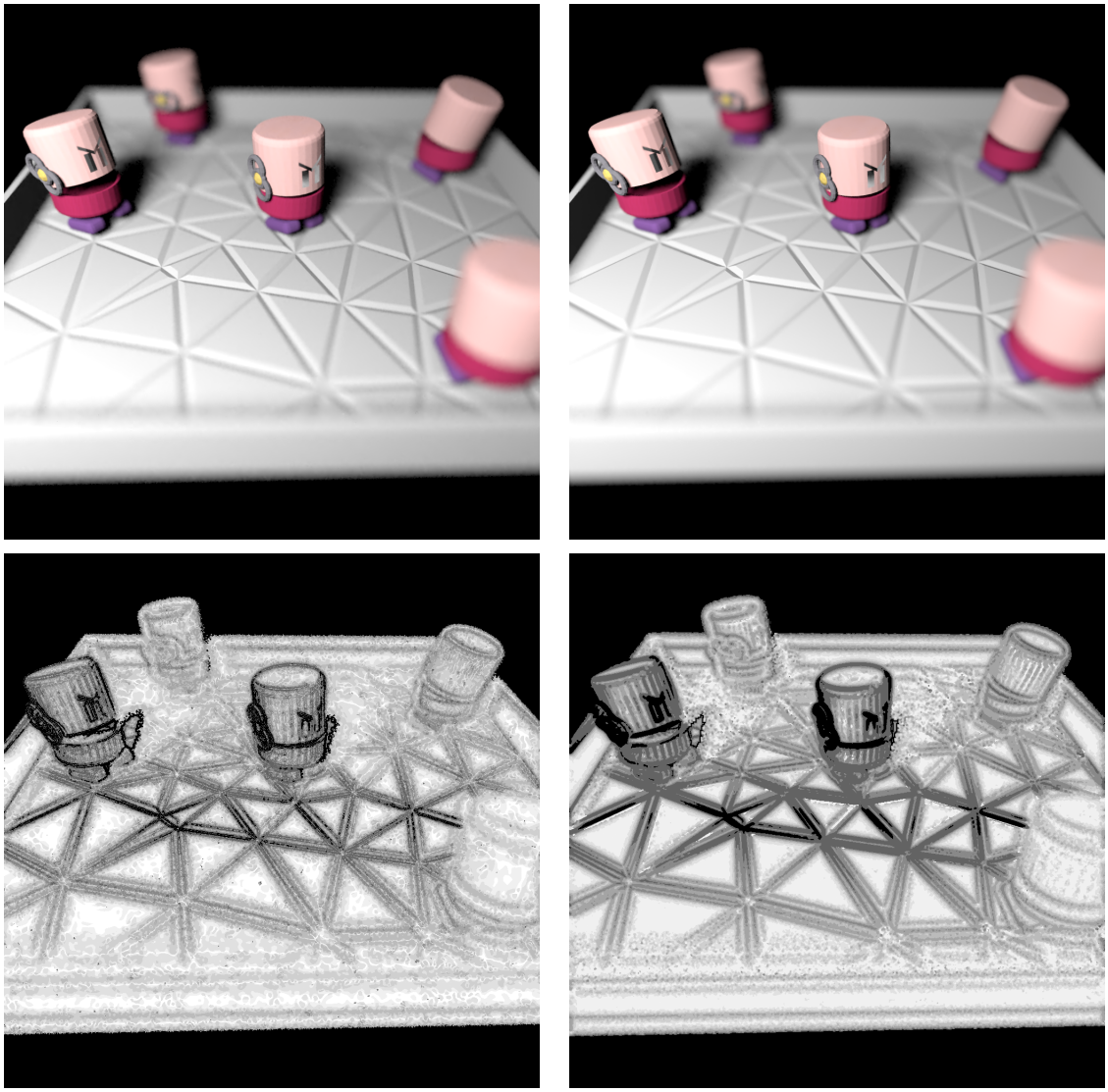
RPF adjusts the weights of cross-bilateral filters by using mutual information and adapts well to scene features in most cases. It also removes the noise produced by few

samples when rendering depth-of-field or motion blur effects. However, its multi-pass reconstruction algorithm can produce slightly oversmoothed results, such as the texture on the floor in *Sibenik* (Figure 5.6), the disappeared shadows in *Sponza* (Figure 5.8), and the glossy reflection on the teapot in *Teapot* (Figure 5.7). Another severe limitation of this approach is that the mutual information must be computed at the sample level, making the computation inefficient in both performance and memory consumption. To render one high-quality image at the 1920x1080 full HD resolution with 64 samples per pixel, it takes up to 13 GB to store the samples (108 bytes per sample as described in the paper). Finally, RPF is designed for reconstruction and does not have a feedback mechanism to the renderer for adaptive sampling.

Our method does away with the limitations of both GEM and RPF. At one end, we adopt SURE to estimate the error of an arbitrary reconstruction kernel. This allows us to optimize over a discrete set of cross bilateral filters for each pixel and determine the optimal sample distribution. Also, we propose a memory-friendly method to detect noisy geometric features when rendering depth of field and motion blur. As a result, our method successfully eliminates MC noise for a wide range of effects while preserving high-frequency textures and fine geometry details.

5.5.4 Other filters

To demonstrate the flexibility of the proposed framework with respect to different filters, in addition to cross bilateral filters, we have also experimented with isotropic Gaussian filters and cross non-local means filters. For isotropic Gaussians, we compare the results with GEM [87] which is specifically designed for optimizing over an isotropic Gaussian filterbank. To be fair, we filter the SURE-estimated MSE using an isotropic Gaussian filter without using scene feature information. As shown in Figure 5.11, results of both methods are comparable and the scale selection maps are similar. This means that



GEM (MSE: 0.000287) Our (MSE: 0.000356)

Figure 5.11: The proposed SURE-based framework incorporated with an isotropic Gaussian filterbank and compared with GEM [87]. The results and scale selection maps generated by both methods are similar.

our SURE optimization is comparable to the specifically-designed GEM for the isotropic case.

The non-local means filter [12] is a popular method for image denoising. It assigns filter weights based on the similarity between pixel neighborhoods. In the context of rendering, we can further utilize scene features for better results. Thus, the cross non-

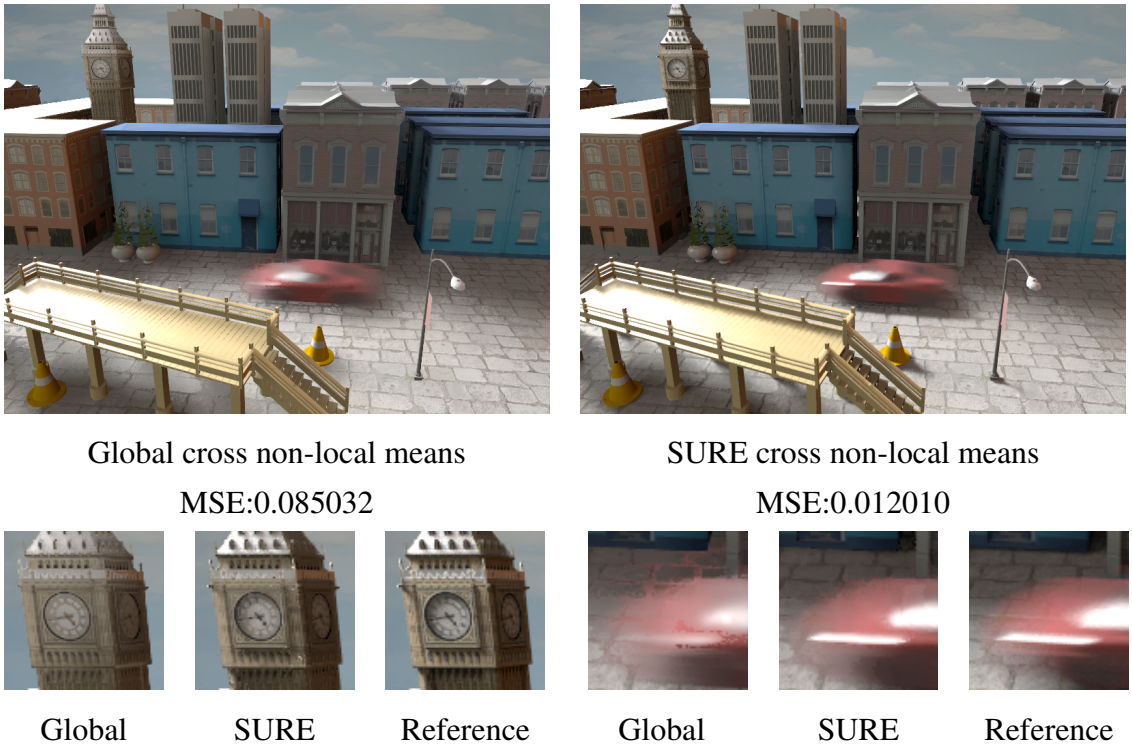


Figure 5.12: Comparison of cross non-local means filters without and with SURE-based framework. Compared to the global cross non-local means filter, our SURE-based optimization largely alleviates the oversmoothing problem. The sampling rate of the noisy input image is about 41 samples per pixel.

local mean filter assigns the weight w_{ij} between two pixels i and j as

$$\exp\left(-\frac{\sum_{n\epsilon N} \parallel c_{i+n} - c_{j+n} \parallel^2}{2|N|{\sigma_r}^2}\right) \prod_{k=1}^{m} \exp\left(-\frac{D(\bar{f}_{ik}, \bar{f}_{jk})^2}{2{\sigma_{f_k}}^2}\right), \tag{5.10}$$

where N is the neighbourhood ($N = \{(x,y)|-2 <= x,y <= 2\}$ in our implementation). Other symbols are the same as defined in Section 5.4.2. Note that we use the patch-based distance only for color information, since patch-based distance for scene features tends to smooth out features. The filtered pixel color $\hat{c}_i$ of pixel i is computed as the weighted combination of the colors c_j of all neighboring pixels j within a 41×41 neighborhood.

To demonstrate the utility of SURE-based filter selection, we applied cross non-local means filters in two settings. For the first setting – the global cross non-local means filter – we used the same range parameter σ_r across the whole image. For the second one – the

SURE cross non-local means filter – we constructed a cross non-local means filterbank by varying σ_r and used SURE to select best filters and shot samples. Figure 5.12 shows the comparisons between the two. It is clear that the SURE-based framework significantly alleviates the over-smoothness problem of the global filter, especially in shadows and in the motion blur of the moving car. From our experiments, filtering with cross non-local means filters sometimes generated slightly better results than cross bilateral filters. However, it is about 10 times slower than the cross bilateral filter. As a compromise between quality and performance, we opted to use cross bilateral filters for most results in the chapter.

5.5.5 Limitations

The *Teapot* scene (Figure 5.7) reveals a limitation of our approach. The bump mapped floor contains a large number of very high-frequency textures. At the same time, it suffers from a large amount of Monte Carlo noise due to the environment lighting and glossy reflection. With a low sample budget, our approach does not preserve all the details well. In addition, as with most reconstruction approaches, our method was susceptible to oversmoothing.

5.6 Conclusion and Future Work

We have presented an efficient adaptive sampling and reconstruction algorithm for reducing noise in Monte Carlo rendering by using Stein's Unbiased Risk Estimator (SURE) in the error estimation framework. For reconstruction, the use of SURE enables us to measure the reconstruction quality for arbitrary filter kernels. It does away with the limitation of using only symmetric kernels imposed by previous work. This freedom to use non-symmetric kernels significantly improves the effectiveness of the framework. When performing adaptive sampling, SURE can be used to determine the sampling den-

sity. Another contribution of this paper is an efficient and memory-friendly approach to detect noisy geometric features when rendering depth of field and motion blur. As a result, the proposed adaptive sampling and reconstruction method efficiently eliminates Monte Carlo noise while preserving the vivid details of a scene. Experiments show the proposed method offers significant improvement over state-of-the-art approaches.

One possible future direction is to implement the proposed algorithms on GPUs for interactive applications. We also would like to extend the SURE-based framework to animation rendering. In the current algorithm, as there is no built-in mechanism specifically designed for temporal data, temporal coherence cannot be guaranteed. In practice, we have experimented with a naive approach that renders each frame independently. The results look good enough with only very subtle temporal flicking. However, a better way to handle animation would be to consider temporal samples and perform filtering in the spatial-temporal domain. Finally, it would also be interesting to adapt SURE to other rendering applications that require error estimation.

Chapter 6

Conclusions

This dissertation consists of three sampling and reconstruction techniques to improve the efficiency of Monte Carlo rendering. First, we proposed VisibilityCluster for efficient visibility approximation. By exploiting the visibility coherence between a cluster of lights and a cluster of shading point, we make it feasible to include the visibility approximation in importance sampling framework. Compared to previous work, noise is significantly reduced due to sampling fewer occluded lights. Our second work focuses on translucent material rendering. We proposed a dual-matrix representation and a sampling and reconstruction framework to reduce irradiance computation for unimportant surface samples, which have little contributions to the final image. The method also accelerates the adaptive diffusion gathering by refining surface areas with large contribution first. Finally, an adaptive sampling and reconstruction framework based on the Stein's Unbiased Risk Estimator (SURE) was proposed to efficiently render distributed effects. By applying SURE into the rendering community, we allow more effective filters to be used for reconstructing samples. Applications of physically-based rendering such as lighting design, scientific visualization, and film production can all be benefit from the above methods.

Appendix A

Derivatives for Filters

To compute SURE for reconstruction filters, we must calculate their derivatives $dF(c_i)/dc_i$ and substitute into Equation 5.2. For the cross bilateral filter, from Equation 5.5, we have (note that $w_{ii} = 1$ according to the definition in Equation 5.4)

$$F(c_i) = \frac{\sum_{j=1}^{n} w_{ij} c_j}{\sum_{j=1}^{n} w_{ij}} = \frac{\sum_{j\neq i} w_{ij} c_j + c_i}{\sum_{j=1}^{n} w_{ij}}. \tag{A.1}$$

Let $W_i = \sum_{j=1}^{n} w_{ij}$. After applying the quotient rule of derivatives, we have

$$\frac{dF(c_i)}{dc_i} = \frac{(\frac{d(\sum_{j\neq i} w_{ij} c_j + 1)}{dc_i}) - \frac{dW_i}{dc_i} F(c_i)}{W_i}. \tag{A.2}$$

After substituting $\frac{dw_{ij}}{dc_i} = \frac{(c_j - c_i)}{\sigma_r^2} w_{ij}$ into Equation A.2, and after some manipulations, we obtain

$$\frac{dF(c_i)}{dc_i} = \frac{1}{W_i} + \frac{1}{\sigma_r^2}(F^2(c_i) - F(c_i)^2), \tag{A.3}$$

where

$$F^2(c_i) = \frac{\sum_{j=1}^{n} w_{ij} {c_j}^2}{\sum_{j=1}^{n} w_{ij}}.$$

For the cross non-local means filter, its $dF(c_i)/dc_i$ is

$$\begin{aligned} \frac{dF(c_i)}{dc_i} = \frac{1}{W_i} + \frac{1}{|N|\sigma_r^2}(F^2(c_i) - F(c_i)^2) + \\ \frac{1}{|N|\sigma_r^2 W_i} \sum_{n\in N} w_{i,i-n}(c_i - c_{i+n})(F(c_i) - c_{i-n}), \end{aligned} \tag{A.4}$$

where W_i and $F^2(c_i)$ are defined similarly as above. Similar derivations can be found in Van De Ville and Kocher's paper [103].

Bibliography

[1] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen. Structured importance sampling of environment maps. *ACM Trans. Graph. (Proc. SIGGRAPH 03)*, 22(3):605–612, July 2003.

[2] M. Agrawala, R. Ramamoorthi, A. Heirich, and L. Moll. Efficient image-based methods for rendering soft shadows. In *Proc. 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH 00)*, pages 375–384, 2000.

[3] A. Arbree. *Scalable And Heterogeneous Rendering Of Subsurface Scattering Materials*. PhD thesis, Cornell University, Oct. 2009.

[4] A. Arbree, B. Walter, and K. Bala. Single-pass scalable subsurface rendering with lightcuts. *Computer Graphics Forum (Proc. Eurographics 08)*, pages 507–516, 2008.

[5] K. Bala, B. Walter, and D. P. Greenberg. Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph. (Proc. SIGGRAPH 03)*, 22(3):631–640, 2003.

[6] F. Banterle and A. Chalmers. A fast translucency appearance model for real-time applications. In *Proc. Spring Conference on Computer Graphics (SCCG 2006)*, April 2006.

[7] T. Bashford-Rogers, K. Debattista, C. Harvey, and A. Chalmers. Approximate visibility grids for interactive indirect illumination. In *Proc. 3rd International Conference on Games and Virtual Worlds for Serious Applications*, pages 55–62, 2011.

[8] P. Bauszat, M. Eisemann, and M. Magnor. Guided image filtering for interactive high-quality global illumination. In *Proc. 22nd Eurographics conference on Rendering (EGSR 11)*, volume 30, pages 1361–1368, 2011.

[9] A. Ben-Artzi, R. Ramamoorthi, and M. Agrawala. Efficient shadows for sampled environment maps. *J. Graphics Tools*, pages 13–36, 2006.

[10] T. Blu and F. Luisier. The SURE-LET approach to image denoising. *IEEE Transactions on Image Processing*, 16(11):2778–2786, 2007.

[11] S. Boulos. How often do shadow rays hit? *Ray Tracing News*, 23(1), July 2010.

[12] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR 05)*, pages 60–65, 2005.

[13] D. Burke, A. Ghosh, and W. Heidrich. Bidirectional importance sampling for direct illumination. In *Proc. 16th Eurographics conference on Rendering Techniques (EGSR 05)*, pages 147–156, 2005.

[14] N. A. Carr, J. D. Hall, and J. C. Hart. GPU algorithms for radiosity and subsurface scattering. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 51–59, 2003.

[15] C.-W. Chang, W.-C. Lin, T.-C. Ho, T.-S. Huang, and J.-H. Chuang. Real-time translucent rendering using gpu-based texture space importance sampling. *Computer Graphics Forum (Proc. Eurographics 08)*, pages 517–526, 2008.

[16] J. Chen, B. Wang, Y. Wang, R. S. Overbeck, J.-H. Yong, and W. Wang. Efficient depth-of-field rendering with adaptive sampling and multiscale reconstruction. *Comput. Graph. Forum*, 30(6):1667–1680, 2011.

[17] Y. Chen, X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum. Shell texture functions. *ACM Trans. Graph. (Proc. SIGGRAPH 04)*, 23(3):343–353, Aug. 2004.

[18] E. Cheslack-Postava, R. Wang, O. Akerlund, and F. Pellacini. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 08)*, 27(5):128:1–128:10, Dec. 2008.

[19] P. H. Christensen, G. Harker, J. Shade, B. Schubert, and D. Batali. Multiresolution radiosity caching for global illumination in movies. In *ACM SIGGRAPH 2012 Talks*, pages 47:1–47:1, 2012.

[20] P. Clarberg and T. Akenine-Möller. Exploiting visibility correlation in direct illumination. In *Proc. 19th Eurographics conference on Rendering (EGSR 08)*, pages 1125–1136, 2008.

[21] P. Clarberg and T. Akenine-Möller. Practical product importance sampling for direct illumination. *Computer Graphics Forum (Proc. Eurographics 08)*, 27(2):681–690, 2008.

[22] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen. Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1166–1175, July 2005.

[23] D. Cline, D. Adams, and P. Egbert. Table-driven adaptive importance sampling. In *Proc. 19th Eurographics conference on Rendering (EGSR 08)*, pages 1115–1123, 2008.

[24] D. Cline, P. K. Egbert, J. F. Talbot, and D. L. Cardon. Two stage importance sampling for direct lighting. In *Proc. 17th Eurographics conference on Rendering Techniques (EGSR 06)*, pages 103–113, 2006.

[25] C. Dachsbacher and M. Stamminger. Translucent shadow maps. In *Proc. 14th Eurographics workshop on Rendering (EGRW 03)*, pages 197–201, 2003.

[26] C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *Proc. Symposium on Interactive 3D graphics and games*, pages 203–231, 2005.

[27] C. Dachsbacher and M. Stamminger. Splatting indirect illumination. In *Proc. Symposium on Interactive 3D graphics and games*, pages 93–100, 2006.

[28] H. Dammertz, D. Sewtz, J. Hanika, and H. P. A. Lensch. Edge-avoiding Á-Trous wavelet transform for fast global illumination filtering. In *Proc. the Conference on High Performance Graphics (HPG 10)*, pages 67–75, 2010.

[29] T. Davidovič, J. Křivánek, M. Hašan, P. Slusallek, and K. Bala. Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 10)*, 29(6):143:1–143:8, Dec. 2010.

[30] L. M. Delves and J. L. Mohamed, editors. *Computational methods for integral equations*. Cambridge University Press, New York, NY, USA, 1986.

[31] E. D'Eon. A better dipole. Technical report, 2012.

[32] E. D'Eon and G. Irving. A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph. (Proc. SIGGRAPH 11)*, 30(4):56:1–56:14, July 2011.

[33] E. d'Eon, D. Luebke, and E. Enderton. Efficient rendering of human skin. In *Proc. 18th Eurographics Conference on Rendering Techniques (EGSR 07)*, pages 147–157, 2007.

[34] Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.-P. Seidel. Real-time indirect illumination with clustered visibility. In *Vision, Modeling, and Visualization Workshop*, 2009.

[35] M. Donikian, B. Walter, K. Bala, S. Fernandez, and D. P. Greenberg. Accurate direct illumination using iterative adaptive sampling. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):353–364, May 2006.

[36] C. Donner and H. W. Jensen. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1032–1039, July 2005.

[37] C. Donner and H. W. Jensen. Rendering translucent materials using photon diffusion. In *Proc. 18th Eurographics conference on Rendering Techniques (EGSR 07)*, pages 243–251, 2007.

[38] D. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224, 1995.

[39] P. Dutré, P. Tole, and D. P. Greenberg. Approximate visibility for illumination computations using point clouds. Technical Report PCG-00-01, Cornell University, 2000.

[40] K. Egan, F. Durand, and R. Ramamoorthi. Practical filtering for efficient ray-traced directional occlusion. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 11)*, 30(6):180:1–180:10, Dec. 2011.

[41] K. Egan, F. Hecht, F. Durand, and R. Ramamoorthi. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph.*, 30(2):9:1–9:13, 2011.

[42] K. Egan, Y.-T. Tseng, N. Holzschuch, F. Durand, and R. Ramamoorthi. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. (Proc. SIGGRAPH 09)*, 28(3):93:1–93:13, 2009.

[43] S. Fernandez, K. Bala, and D. P. Greenberg. Local illumination environments for direct lighting acceleration. In *Proc. 13th Eurographics workshop on Rendering*, pages 7–14, 2002.

[44] C. V. Fiorio. Confidence intervals for kernel density estimation. *Stata Journal*, 4(2):168–179, 2004.

[45] I. Georgiev, J. Křivánek, S. Popov, and P. Slusallek. Importance caching for complex illumination. *Computer Graphics Forum (Proc. Eurographics 12)*, 31(2):701–710, May 2012.

[46] A. Ghosh and W. Heidrich. Correlated visibility sampling for direct illumination. *Visual Computer*, 22(9):693–701, Sept. 2006.

[47] Y. Gong, W. Chen, L. Zhang, Y. Zeng, and Q. Peng. GPU-based rendering for deformable translucent objects. *Vis. Comput.*, 24(2):95–103, Jan. 2008.

[48] R. Habel, P. H. Christensen, and W. Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. In *Proc. 24th Eurographics conference on Rendering Techniques (EGSR 13)*, 2013.

[49] T. Hachisuka, W. Jarosz, and H. W. Jensen. A progressive error estimation framework for photon density estimation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 10)*, 29(6):144:1–144:12, 2010.

[50] T. Hachisuka, W. Jarosz, R. P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. W. Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. (Proc. SIGGRAPH 08)*, 27(3):33:1–33:10, 2008.

[51] X. Hao and A. Varshney. Real-time rendering of translucent meshes. *ACM Trans. Graph.*, 23(2):120–142, Apr. 2004.

[52] D. Hart, P. Dutré, and D. P. Greenberg. Direct illumination with lazy visibility evaluation. In *Proc. 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH 99)*, pages 147–154, 1999.

[53] M. Hašan, J. Křivánek, B. Walter, and K. Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 09)*, 28(5):143:1–143:6, Dec. 2009.

[54] M. Hašan, F. Pellacini, and K. Bala. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. (Proc. SIGGRAPH 07)*, 26(3), July 2007.

[55] T. Ize and C. D. Hansen. RTSAH traversal order for occlusion rays. In *Computer Graphics Forum (Proc. Eurographics 11)*, volume 30, pages 297–305, 2011.

[56] W. Jarosz, N. A. Carr, and H. W. Jensen. Importance sampling spherical harmonics. *Computer Graphics Forum (Proc. Eurographics 09)*, 28(2):577–586, Apr. 2009.

[57] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph. (Proc. SIGGRAPH 02)*, 21(3):576–581, July 2002.

[58] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proc. 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 01)*, pages 511–518, 2001.

[59] J. T. Kajiya. The rendering equation. In *Proc. 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH 86)*, pages 143–150, 1986.

[60] A. Keller. Instant radiosity. In *Proc. 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH 97)*, pages 49–56, 1997.

[61] S.-L. Keng, W.-Y. Lee, and J.-H. Chuang. An efficient caching-based rendering of translucent materials. *Vis. Comput.*, 23(1):59–69, Dec. 2006.

[62] A. King, C. Kulla, A. Conty, and M. Fajardo. BSSRDF importance sampling. In *ACM SIGGRAPH 2013 Talks*, pages 48:1–48:1, 2013.

[63] T. Kollig and A. Keller. Illumination in the presence of weak singularities.

[64] T. Kollig and A. Keller. Efficient illumination by high dynamic range images. In *Proc. 14th Eurographics workshop on Rendering*, pages 45–50, 2003.

[65] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient BRDF importance sampling using a factored representation. *ACM Trans. Graph. (Proc. SIGGRAPH 04)*, 23(3):496–505, Aug. 2004.

[66] J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph. (Proc. SIGGRAPH 11)*, 30(4):55:1–55:12, 2011.

[67] J. Lehtinen, T. Aila, S. Laine, and F. Durand. Reconstructing the indirect light field for global illumination. *ACM Trans. Graph. (Proc. SIGGRAPH 12)*, 31(4):51:1–51:10, 2012.

[68] J. Lehtinen, M. Zwicker, E. Turquin, J. Kontkanen, F. Durand, F. X. Sillion, and T. Aila. A meshless hierarchical representation for light transport. *ACM Trans. Graph. (Proc. SIGGRAPH 08)*, 27(3):37:1–37:9, Aug. 2008.

[69] H. P. A. Lensch, M. Goesele, P. Bekaert, J. Kautz, M. A. Magnor, J. Lang, and H.-P. Seidel. Interactive rendering of translucent objects. In *Proc. 10th Pacific Conference on Computer Graphics and Applications (PG 02)*, 2002.

[70] H. Li, F. Pellacini, and K. E. Torrance. A hybrid monte carlo method for accurate and efficient subsurface scattering. In *Proc. Sixteenth Eurographics conference on Rendering Techniques (EGSR 05)*, pages 283–290, 2005.

[71] T. Mertens, J. Kautz, P. Bekaert, F. Van Reeth, and H.-P. Seidel. Efficient rendering of local subsurface scattering. In *Proc. 11th Pacific Conference on Computer Graphics and Applications (PG 03)*, 2003.

[72] D. P. Mitchell. Generating antialiased images at low sampling densities. In *Proc. 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH 87)*, pages 65–72, 1987.

[73] D. P. Mitchell. Spectrally optimal sampling for distribution ray tracing. In *Proc. 18th annual conference on Computer graphics and interactive techniques (SIGGRAPH 91)*, pages 157–164, 1991.

[74] A. Muñoz, J. I. Echevarria, F. J. Serón, and D. Gutierrez. Convolution-based simulation of homogeneous subsurface scattering. *Comput. Graph. Forum*, 30(8):2279–2287, 2011.

[75] S. G. Narasimhan, M. Gupta, C. Donner, R. Ramamoorthi, S. K. Nayar, and H. W. Jensen. Acquiring scattering properties of participating media by dilution. *ACM Trans. Graph. (Proc. SIGGRAPH 06)*, 25(3):1003–1012, July 2006.

[76] G. Nichols and C. Wyman. Multiresolution splatting for indirect illumination. In *Proc. Symposium on Interactive 3D graphics and games*, pages 83–90, 2009.

[77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Radiometry. chapter Geometrical considerations and nomenclature for reflectance, pages 94–145. 1992.

[78] J. Novák, T. Engelhardt, and C. Dachsbacher. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. Symposium on Interactive 3D graphics and games*, pages 119–124, 2011.

[79] J. Ou. *Sampling for Complexity in Rendering*. PhD thesis, Dartmouth college, Hanover, New Hampshire, May 2013.

[80] J. Ou and F. Pellacini. LightSlice: matrix slice sampling for the many-lights problem. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 11)*, 30(6):179:1–179:8, Dec. 2011.

[81] R. S. Overbeck, C. Donner, and R. Ramamoorthi. Adaptive wavelet rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 09)*, 28(5):140:1–140:12, 2009.

[82] M. Pharr and G. Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[83] R. Ramamoorthi, J. Anderson, M. Meyer, and D. Nowrouzezahrai. A theory of monte carlo visibility sampling. *ACM Trans. Graph.*, 31(5):121:1–121:16, Sept. 2012.

[84] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 08)*, 27(5):129:1–129:8, Dec. 2008.

[85] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating dynamic global illumination in image space. In *Proc. Symposium on Interactive 3D graphics and games*, pages 75–82, 2009.

[86] F. Rousselle, P. Clarberg, L. Leblanc, V. Ostromoukhov, and P. Poulin. Efficient product sampling using hierarchical thresholding. *Visual Computer*, 24(7):465–474, July 2008.

[87] F. Rousselle, C. Knaus, and M. Zwicker. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 11)*, 30(6):159:1–159:12, 2011.

[88] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *Proc. 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, pages 53–60, 2006.

[89] P. Sen and S. Darabi. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.*, 31(3):18:1–18:15, 2012.

[90] M. A. Shah, J. Konttinen, and S. Pattanaik. Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE Comput. Graph. Appl.*, 29(1):66–78, Jan. 2009.

[91] Y. Sheng, Y. Shi, L. Wang, and S. G. Narasimhan. A practical analytic model for the radiosity of translucent scenes. In *Proc. symposium on Interactive 3D graphics*, 2013.

[92] P. Shirley, T. Aila, J. Cohen, E. Enderton, S. Laine, D. Luebke, and M. McGuire. A local image reconstruction algorithm for stochastic rendering. In *Proc. Symposium on Interactive 3D graphics and games*, pages 9–14, 2011.

[93] P. Shirley, C. Wang, and K. Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Trans. Graph.*, 15(1):1–36, Jan. 1996.

[94] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. (Proc. SIGGRAPH 03)*, 22(3):382–391, July 2003.

[95] P.-P. Sloan, B. Luna, and J. Snyder. Local, deformable precomputed radiance transfer. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1216–1224, July 2005.

[96] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. Sillion. Fourier depth of field. *ACM Trans. Graph.*, 28(2):18:1–18:12, 2009.

[97] C. M. Stein. Estimation of the mean of a multivariate normal distribution. *Annals of Statistics*, 9(6):1135–1151, 1981.

[98] J. Steinhurst and A. Lastra. Global importance sampling of glossy surfaces using the photon map. In *Proc. IEEE Symposium on Interactive Ray Tracing*, pages 133–138, 2006.

[99] A. J. Stewart and T. Karkanis. Computing the approximate visibility map, with applications to form factors and discontinuity meshing. In *Proc. 9th Eurographics workshop on Rendering*, pages 57–68.

[100] J. F. Talbot, D. Cline, and P. Egbert. Importance resampling for global illumination. In *Proc. 16th Eurographics conference on Rendering Techniques (EGSR 05)*, pages 139–146, 2005.

[101] R. Tamstorf and H. W. Jensen. Adaptive sampling and bias estimation in path tracing. In *Proc. 8th Eurographics workshop on Rendering*, pages 285–295, 1997.

[102] X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum. Modeling and rendering of quasi-homogeneous materials. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1054–1061, July 2005.

[103] D. Van De Ville and M. Kocher. SURE-based non-local means. *IEEE Signal Processing Letters*, 16(11):973–976, 2009.

[104] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, 1998.

[105] E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Proc. 5th Eurographics workshop on Rendering", pages = 147–162, month = Jun, year = 1994,.*

[106] B. Walter, A. Arbree, K. Bala, and D. P. Greenberg. Multidimensional lightcuts. *ACM Trans. Graph. (Proc. SIGGRAPH 06)*, 25(3):1081–1088, July 2006.

[107] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1098–1107, July 2005.

[108] B. Walter, P. Khungurn, and K. Bala. Bidirectional lightcuts. *ACM Trans. Graph. (Proc. SIGGRAPH 12)*, 31(4):59:1–59:11, July 2012.

[109] R. Wang and O. Åkerlund. Bidirectional importance sampling for unstructured direct illumination. *Computer Graphics Forum (Proc. Eurographics 09)*, 28(2):269–278, Apr. 2009.

[110] R. Wang, J. Tran, and D. Luebke. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. (Proc. SIGGRAPH 05)*, 24(3):1202–1207, July 2005.

[111] R. Xu and S. N. Pattanaik. A novel Monte Carlo noise reduction operator. *IEEE Computer Graphics and Applications*, 25(2):31–35, 2005.

[112] L.-Q. Yan, Y. Zhou, K. Xu, and R. Wang. Accurate translucent material rendering under spherical gaussian lights. *Comp. Graph. Forum*, 31(7pt2):2267–2276, Sept. 2012.

[113] I. Yu, A. Cox, M. H. Kim, T. Ritschel, T. Grosch, C. Dachsbacher, and J. Kautz. Perceptual influence of approximate visibility in indirect illumination. *ACM Trans. Appl. Percept.*, 6(4):24:1–24:14, Oct. 2009.