TU Delft

Delft University of Technology

Alternative Representations and Techniques for Accelerated Realistic Image Synthesis
Improved Sampling, Reconstruction and Modelling Methods

Guo, Jerry Jinfeng

DOI
10.4233/uuid:5f127a15-5fa4-4eae-94fb-d0f30bdebe8c

Publication date
2022

Document Version
Final published version

Citation (APA)
Guo, J. J. (2022). *Alternative Representations and Techniques for Accelerated Realistic Image Synthesis: Improved Sampling, Reconstruction and Modelling Methods*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:5f127a15-5fa4-4eae-94fb-d0f30bdebe8c

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

JERRY JINFENG GUO

ALTERNATIVE REPRESENTATIONS AND TECHNIQUES FOR ACCELERATED REALISTIC IMAGE SYNTHESIS

ALTERNATIVE REPRESENTATIONS AND TECHNIQUES FOR ACCELERATED REALISTIC IMAGE SYNTHESIS

Improved Sampling, Reconstruction and Modelling Methods



Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
vrijdag 18 november 2022 om 12:30 uur

door

Jinfeng GUO

Master of Science Informatica,
Universiteit Utrecht, Nederland,
geboren te Rizhao, China.

Dit proefschrift is goedgekeurd door de

promotoren: Prof. dr. E. Eisemann
copromotor: Dr. M.J. Billeter

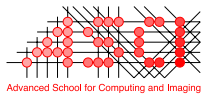
Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. E. Eisemann,	Technische Universiteit Delft
Dr. M.J. Billeter,	University of Leeds, UK
Onafhankelijke leden:	
Prof. dr. J. Dik,	Technische Universiteit Delft
Prof. dr. C. Dachsbacher,	KIT, Germany
Prof. dr. S. Lee,	Sungkyunkwan University, South Korea
Dr. K. Subr,	University of Edinburgh, UK
Dr. A. Weidlich,	NVIDIA Research, Canada
Prof. dr. G. Smaragdakis,	Technische Universiteit Delft, reservelid

The work in the thesis has been carried out under the auspices of the research school ASCI (Advanced School for Computing and Imaging). ASCI dissertation number 442.







Printed by: Proefschrift Specialist

Cover: SemiProWinner

Copyright ©2022 by J. Guo
ISBN 978-94-6384-394-2

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

This thesis is dedicated to my family.

SUMMARY

Realism has always been a major goal in visual content creation - from oil painting to motion pictures, from graphic arts to scientific data visualization. Computer graphics creates a virtual reality with digital representations. Trading between accuracy and speed, realistic rendering either creates photorealistic renders that follow strict rules of physics or approximates them with interactive alternatives.

The two approaches have their distinctive strengths and constraints. In this thesis, we focus on working from both ends towards realistic rendering. We study the highly accurate process of physically based rendering and introduce three novel methods dedicated to making it more efficient. We also investigate real-time depth of field rendering and develop a new model for an effect that is currently missing in state of the art systems.

Part i concerns sampling in numerical integration. Two methods are presented in Chapter 2 and Chapter 3. We first propose to perform path guiding in primary sample space, resulting in an effective and efficient scheme that is easy to plug into existing rendering pipelines. Secondly, we map visibility relations in a matrix-like table to steer the sampling process, which improves processes, such as visible light samples and light subpaths.

Part ii tackles the subsequent step after sampling in numerical integration. We present a new integration scheme in Chapter 4 that associates a weight to samples based on their adjacency, while remaining unbiased. The method delivers similar performance using uniform random samples as one can obtain with costly low-discrepancy sequences.

In Part iii this thesis revisits the optics behind depth of field effects and models distortion and shrinking effects that are missing in modern real-time rendering solutions. We are able to deliver similar effects to that of ray-traced results at a fraction of the cost.

Each chapter has detailed depiction and evaluation that helps readers better understand the methods and gain insights as to the applications thereof.

SAMENVATTING

Realisme is altijd een belangrijk doel geweest bij het creëren van visuele media - van olieverfschilderijen tot speelfilms, van grafische kunst tot de visualisatie van wetenschappelijke gegevens. Computer graphics creëert een virtuele realiteit door middel van digitale representaties. Realistische rendering maakt een afweging tussen nauwkeurigheid en snelheid en produceert ofwel fotorealistische renders die de strikte regels van de fysica volgen, of maakt gebruik van interactieve alternatieven die deze regels benaderen.

De twee methodes hebben beide hun eigen sterktes en beperkingen. In deze dissertatie werken we vanuit beide kanten naar het doel van realistische rendering. We bestuderen het zeer nauwkeurige proces van fysisch gebaseerde rendering en introduceren drie nieuwe methoden om het efficiënter te maken. Ook onderzoeken we real-time scherptediepte rendering en ontwikkelen een nieuw model voor een effect dat momenteel ontbreekt in de meest geavanceerde systemen.

Teil i betreft bemonstering in numerieke integratie. Twee methoden worden gepresenteerd in Kapitel 2 en Kapitel 3. In de eerste methode dragen we aan om padgeleiding uit te voeren in de primaire bemonsteringsruimte, wat resulteert in een effectief en efficiënt model dat gemakkelijk kan worden toegevoegd aan bestaande renderpijplijnen. In de tweede methode brengen we zichtbaarheidsrelaties in kaart in een matrixachtige tabel om het bemonsteringsproces te sturen, wat processen voor zichtbare lichtmonsters en lichtsubpaden verbetert.

Teil ii pakt de volgende stap na bemonstering in numerieke integratie aan. We presenteren een nieuw integratiemodel in Kapitel 4 dat een gewicht toekent aan monsters op basis van hun nabijheid, en toch onpartijdig blijft. De methode levert met uniforme willekeurige monsters vergelijkbare prestaties als dure lage-discrepantie reeksen.

In Teil iii bekijken we de optica achter scherptediepte-effecten opnieuw en worden vervorming en krimpeffecten gemodelleerd die ontbreken in moderne real-time renderoplossingen. We zijn in staat om vergelijkbare effecten te produceren als met raytraced resultaten tegen een fractie van de kosten.

Elk hoofdstuk heeft gedetailleerde weergaves en evaluaties die de lezer helpt de methoden beter te begrijpen en inzicht te krijgen in de toepassingen ervan.

CONTENTS

List of Figures xii
List of Tables xiii
Listings xiii

1 Introduction 1
1.0.1 Proposed Techniques 4
1.0.2 Contributions 6

I Sampling
2 PSSPG 11
2.1 Introduction 14
2.2 Related Work 15
2.2.1 Distinction 18
2.3 Method 18
2.3.1 Formulation 18
2.3.2 Overview 19
2.3.3 k-d tree 21
2.3.4 Tree Construction 21
2.3.5 Tree Sampling 22
2.3.6 Rendering 23
2.4 Implementation and Results 24
2.4.1 Implementation 24
2.4.2 Comparing Standards 24
2.4.3 Results 25
2.5 Discussion 30
2.6 Conclusion and Future Work 30
3 NEE++ 33
3.1 Introduction 36
3.2 Related Work 37
3.3 Visibility Mapping 39
3.3.1 Data Structure 39
3.3.2 Obtaining Visibility Information 39
3.3.3 Using Visibility Information 41
3.4 Applications of Visibility Mapping 42
3.4.1 NEE++ Rejection Sampling Shadow Test 43
3.4.2 NEE++ Direct Light Sampling Strategy 43
3.4.3 NEE++ Bi-Directional Subpath Sampling 43
3.5 Results 45
3.5.1 Experiment Objectives 46
3.5.2 NEE++ Rejection Sampling Shadow Test 47

3.5.3 NEE++ Direct Light Sampling Strategy 48
3.5.4 NEE++ Bi-Directional Subpath Sampling 53
3.6 Discussions 55
3.6.1 Grid resolution 59
3.6.2 Compatibility with Path Guiding 59
3.6.3 Comparison with PCBPT 60
3.6.4 Limitations 60
3.6.5 Future work 60
3.7 Conclusions 61

II Reconstruction
4 GSR4MC 65
4.1 Introduction 68
4.2 Background 69
4.3 Formulation and Problem Statement 71
4.4 Geometric Sample Reweighting 71
4.4.1 Consistent Estimator 71
4.4.2 Bias Analysis 72
4.4.3 Deriving an Unbiased Estimator 74
4.4.4 Working with Non-uniform Samples 75
4.5 Results 75
4.5.1 Numerical Performance 76
4.5.2 Application to Rendering 79
4.6 Discussion 85
4.7 Conclusion 85

III Modelling
5 Effective CoC 93
5.1 Introduction 96
5.2 Background 96
5.2.1 Prior Work 100
5.3 Our method 102
5.3.1 Effective Circle of Confusion 103
5.3.2 Pipeline Overview 104
5.3.3 Importance Function for Lens/Aperture Sampling 107
5.4 Implementations and Results 108
5.4.1 Effective CoC for Real-time DoF with Layered Rendering 110
5.4.2 Aperture Sampling 113
5.5 Conclusions 120
6 Conclusion 121

Appendix
A Appendix 127

A.1 Expectation of integration over given sample set 127
A.2 Expectation of integration with augmented periodic samples 128

Bibliography 129
Curriculum Vitae 139
List of Publications 141
Acknowledgments 143

LIST OF FIGURES

Figure 1.1 Real-time ray tracing 2
Figure 1.2 Unreal engine in production 2
Figure 2.1 Path construction 18
Figure 2.2 PSSPG pipeline 19
Figure 2.3 PSSPG sample visualization 22
Figure 2.4 The *Ajar* scene. 26
Figure 2.5 The *Dining room* scene. 27
Figure 2.6 The *staircase* scene 27
Figure 2.7 The *kitchen* scene. 28
Figure 2.8 Convergence plot of all our tests. 28
Figure 3.1 NEE++ Overview 40
Figure 3.2 Visibility map example 41
Figure 3.3 NEE++ in BDPT 46
Figure 3.4 NEE++ rejection sampling 49
Figure 3.5 NEE++ Direct Light Sampling 51
Figure 3.6 Equal rMSE comparison 51
Figure 3.7 *Staircase* scene 52
Figure 3.8 NEE++ bi-directional light sampling - *Test Corridor* scene 54
Figure 3.9 Statistics of the *Test Corridor* scene 55
Figure 3.10 *Nasty Classroom* scene 56
Figure 3.11 Equal quality comparison NEE++ bi-directional light sampling 56
Figure 3.12 The *Bathroom* scene 57
Figure 3.13 Voxel-grid resolutions 57
Figure 4.1 Illustration of sampling and reconstruction 68
Figure 4.2 Numerical performance of reweighting both uniform and two non-uniform distributions 77
Figure 4.3 GSR numerical tests 78
Figure 4.4 GSR convergence 1D functions 80
Figure 4.5 GSR for motion blur 1 81
Figure 4.6 GSR for motion blur 2 81
Figure 4.7 GSR for spectral rendering 1 82
Figure 4.8 GSR for spectral rendering 1 83
Figure 4.9 GSR for spectral defocus blur 84
Figure 4.10 GSR for light sampling 1 87
Figure 4.11 GSR for light sampling 2 88
Figure 4.12 GSR for light sampling 3 89
Figure 5.1 Classic lens model 97
Figure 5.2 Size of circle of confusion 98

Figure 5.3 Occulusion-induced changes to CoC 99
Figure 5.4 CoC as a function of depth 100
Figure 5.5 Simplified perspective camera model 101
Figure 5.6 CoC lensing and distortion phenomenon 103
Figure 5.7 Simplified illustration of *effective CoC* 1 105
Figure 5.8 Effective CoC pipeline overview 106
Figure 5.9 Illustration of *effective CoC* 2 106
Figure 5.10 Maintaining magic vector 108
Figure 5.11 Lens sampling illustration 109
Figure 5.12 Three layer rendering setup 110
Figure 5.13 Composition demonstration in the *Sponza* scene 113
Figure 5.14 Views of the *Sponza* scene 114
Figure 5.15 Views of the *Bistro interior* scene 115
Figure 5.16 The modified *Kitchen* scene 116
Figure 5.17 The modified *Classroom* scene 116
Figure 5.18 The modified *Staircase* scene 117

LIST OF TABLES

Table 2.1 Zero radiance path ratios 29
Table 3.1 Tests timing 48
Table 3.2 NEE++ statistics 50
Table 3.3 NEE++ as light sampling strategy - *Staircase* scene runtime 50
Table 3.4 NEE++ as light sampling strategy - *Staircase* scene statistics 52
Table 3.5 NEE++ for Bi-Directional subpath sampling statistics 58
Table 5.1 Runtime breakdown of tests 118
Table 5.2 Lens sampling statistics 119

LISTINGS

1
Horizontal pass shader code for near layer112

1 INTRODUCTION

远看山有色，近听水无声。春去花还在，人来鸟不惊。 *Poetry by* Wang Wei *(699-759 AD), a poet and painter from* Tang *dynasty, vidvidly describes the level of realism and immersion of a particular painting.*

Within computer science, *computer graphics* is a vast research field focusing on the theory, computation, engineering and application of visual content. *Image synthesis*, commonly referred to as *rendering*, can serve as a lens between human users and virtual descriptions inside computer systems. Rendering refers to the process that turns digital scenes, such as shapes, lighting conditions, material appearance and virtual cameras, into images. *Realistic rendering*, as the name suggests, strives to deliver visual content that is as close to reality as possible.

With detailed digital descriptions, either artist-crafted, machine-captured or procedural-generated, rendering algorithms simulate a virtual reality. There are two schools of realistic rendering depending on the application context: offline and real-time (or online). The former tries to follow strict physics models and simulate light-matter interaction often as precisely as possible. The results are usually photorealistic images that are commonly seen in motion picture productions or for predictive rendering in architecture, which means that these images can be prepared in a lengthy process. Real-time rendering targets highly interactive applications such as video games or general human-machine interaction. Contrary to an offline setting, images cannot be precomputed, as they depend on just-in-time input. This high-performance requirements often imply a sacrifice with respect to the precision of the simulation.

Just like many other subjects in computer science, both offline and real-time rendering benefit from evolution in software as well as hardware. Faster hardware increases the throughput of operations, which means that not only previously considered offline effects find their way into real-time applications but also more and more techniques native to a real-time context are developed towards increasing realism. The same holds for offline methods that can be increasingly precise due to the algorithmic evolution and more efficient performance-wise, benefiting from, e.g., real-time techniques. Thus, the evolution of virtual realism is accelerated not only by the ever-increasing compute power from hardware, but also through the advent of more advanced rendering algorithms and computational models.

As witnessed, recent developments in graphics has brought new opportunities to the horizon. Traditional offline rendering solutions depend heavily on physically-based path tracing, which builds on top of ray-tracing operations. Graphics hardware (*graphics processing*

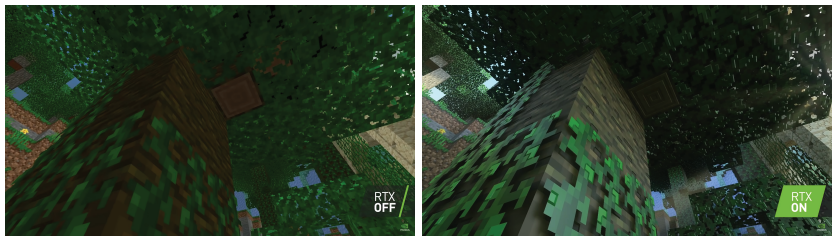


Figure 1.1: Game *Minecraft* with (left) and without (right) ray tracing effects. Notice the added realism of ray traced lighting effects. Image courtesy of *NVIDIA*.

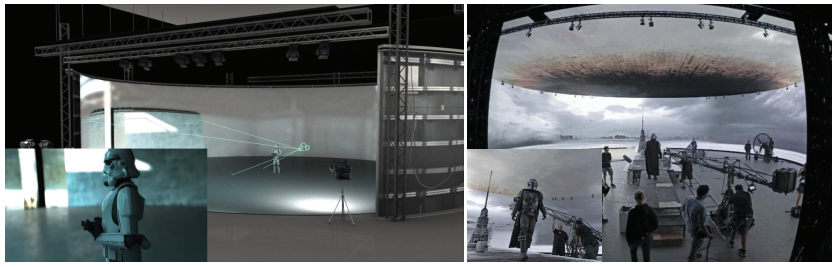


Figure 1.2: Production of *Mandalorian* using *Unreal engine 4*. The background of the footage, which uses real-time renderings from Unreal engine, matches the exact movement and angle of the camera. Image courtesy of *Industrial Light & Magic* and *Unreal*.

unit, GPU), which is designed for extremely parallelizable tasks, on its own is becoming increasingly sophisticated. With the introduction of hardware-accelerated ray tracing combined with high processing power, the line between offline and real-time rendering is starting to blur. This is evident in modern games featuring ray-traced effects and the adoption of game engines in film productions. Some examples include ray-traced reflections in *Battlefield V* and path traced global illumination in *Minecraft RTX* (see Figure 1.1). Both effects are commonly approximated with various real-time techniques (e.g., screen space techniques) before the introduction of respective offline solutions. The production of *Mandalorian* utilize the realistic rendering tool that is integrated into shooting in real-time (see Figure 1.2).

To understand what makes the realistic rendering problem a challenge, let us first go back to the common ground of both offline and real-time rendering. In light-transport simulation theory, an image is modeled as the integrated radiant power captured by a virtual camera after light-object interaction. Each pixel of this image is evaluated as an integral of incoming light, after it is emitted from light sources, travels through space, and interacts with objects (surface or medium) in the form of reflection, refraction, absorption, etc. When light sources

continuously emit radiant energy, the distribution of radiance within the scene reaches an equilibrium. This very equilibrium is the solution that light-transport algorithms are after. In other words, rendering is equivalent to taking an instant snapshot of a virtual scene where light is fully distributed. From this point, the two rendering schools take different approaches in methodology.

State of the art offline solutions like path tracing send rays from the camera instead of light sources. The terminology path tracing comes from the fact that such rays are recursively traced, forming a connected path consisting of one or more path vertices. The opposite way of sending measurement rays (camera rays), instead of sending spectrum energy, is typically more efficient as a vast majority of the light distribution in the scene is not recorded by the observer.

The Rendering Equation is the mathematical description of the integration process [7]. The rendering equation includes all direct and indirect illumination and compactly describes the underlying high-dimensional integration problem. *Monte Carlo integration* enjoys extensive adoption in offline rendering due to its dimensionality invariance and helps us to approximate the result of this equation numerically.

The difficulties in offline rendering usually lie within its fundamental building blocks. Among others, stochastic variance of numerical integration is often the bottleneck as the convergence behavior is directly reflected in final output in the form of noise at certain level. Such level of noise typically drops to acceptable threshold, subject to artistic purposes, given enough time. Despite the dimensionality invariance property of Monte Carlo integration techniques, the rate of convergence still depends largely on the problem, i.e., the integrand. For instance, in the production of the 2017 CG film *Coco*, an averaged 450 hours per frame is spent on rendering alone. Variance reduction is therefore a major research direction in offline rendering.

On the real-time rendering side, scene geometries, typically triangles, are rasterized into an image. Programmable shaders allow to intervene in this pipeline to influence the result. Different from path tracing, some effects such as reflection and shadowing are not native to rasterization. Dedicated models are typically developed for specific effects, such as making use of screen-space information for reflection and pre-computing object-to-light distance for shadows. The effect-specific modeling for efficient rendering is often where the difficulty of real-time lies, as accurate processes are out-of-reach and the extent of approximation versus cost is the major trade-off. It is therefore a fundamental research direction in real-time rendering to develop new models and appropriate schemes for certain effects. One such problem of concern in this thesis is the depth of field effect, which is a visual depth cue that builds on top of defocus blur.

This thesis is dedicated to exploring the possibility of improving both ends of the realistic-rendering spectrum. Addressing the aforementioned difficulties, we propose new algorithms to make offline rendering deliver realistic results faster. This is achieved with smarter sample placement and reconstruction for numerical integration. We develop new a model that adds a more realistic look in real-time depth of field rendering while maintaining high performance. With this big picture, we zoom-in and inspect the pipelines of both offline and real-time rendering. This is done with three parts in this thesis: sampling, reconstruction and modeling.

1.0.1 *Proposed Techniques*

In Part i, we work with sampling techniques for Monte Carlo integration. This involves targeting the specific steps in the light-transport simulation. To give a short background, typically a connected path is constructed in one out of two ways. The first is that we sample a scattered direction at an intersection point to determine, where the measurement ray would go next. This is done recursively until a light emitting object is hit. Light energy is subsequently transported backwards, scaled by the throughput that is dictated by scattering events at all path vertices. The second approach explicitly checks if a sample point on a light source and the current scattering interaction point are mutually visible [19]. Once a visible pair is available, light energy could flow directly through this newly connected path.

Chapter 2 introduces a mechanism to influence the choice of the aforementioned path construction. The solution relates to *path guiding*, which is usually done by placing directional cache records in the spatial domain This complicated design, while effective, usually makes it challenging to plug into an existing rendering pipelines.

We propose *primary sample space path guiding* that does not require explicitly maintaining and querying of the spatial-directional cache records. Primary sample are random numbers, each sample in this space is therefore a vector of values. We maintain a data structure to encode prior knowledge that is gathered in a pilot pass. Here, we record for the sample vectors the resulting contribution to be able to influence future samples. We then work outside the rendering kernel and take the high dimensional sample vector as input, whose values are subsequently transformed based on our data structure.

Chapter 3 introduces a new perspective to light sampling. During path construction, light sources are constantly sampled, yielding shadow rays. The visibility need to be tested by checking ray-object intersection with other geometries in scene. This sampling process typically relies on information such as the physical area of the light, its emitting power, etc. Samples generated based on this information, tend

to be proportional to the potential contribution which still needs to be scaled by the binary visibility term. In complex scenes, this can become a problem as occluded light samples do not actually contribute any energy and a significant amount of expensive shadow test operations are wasted.

We propose *next event estimation++*, a new way of looking at visibility tests. We map accumulated spatial-spatial visibility information onto a symmetric matrix-like structure, which we call the visibility map. The scene is decomposed into voxel units and each entry in the map records a floating point value, approximating the visibility between two voxels. This information can even be generated during normal path tracing and visibility testing, making collecting it cheap in terms of resources. With this visibility map, we can then improve on light sampling in three ways.

First, given a sample point, we can make an educated guess how much this sample could contribute based on the stored visibility. We can subsequently carry out the visibility test only if the estimation to contribute is high.

Second, when sampling light sources, we can incorporate information in the map such that the generated light samples have a better chance of contributing.

Third, in *bi-directional path tracing* rendering solutions, two subpaths, respectively from the camera and light sources, are connected when possible, which implies a test for each vertex combination. Our visibility map can be utilized here as well. Each camera-light subpath pair could be queried, resulting in a distribution that is proportional to visibility probabilities of bi-directional connections.

In Part ii, we revisit the fundamental process of general Monte Carlo integration. After samples have been generated and evaluated, sample contributions need to be combined in a way that is both unbiased and efficient. Given a certain sample input, the standard way is to use the weighted average of all sample evaluations. The weights are typically associated with a sample probability density. We present a novel geometric perspective of looking at this reconstruction phase in Chapter 4. We start with the assumption of working with a fixed sample set. We demonstrate via a reformulation of the integration process that standard weights fail to faithfully represent each sample's importance, as gaps and overlaps between samples are not taken into consideration. We propose to make use of the adjacency information of samples to determine per-sample weights. A thorough derivation is provided for the correct way to adjust the geometric weights such that the result remains unbiased. A numerical analysis shows our reconstruction method applied to uniform random samples achieves similar convergence behavior as low-discrepancy sequences. Our method contributes to path-tracing applications in the context of temporal sampling, spec-

tral sampling, lens sampling and light sampling show the benefit of using our geometric sample weights.

Just like in the offline domain, many effects in real-time rendering also require integrating results over certain domain, e.g., temporal or lens. Lens sampling typically relates to the integration process with respect to the open aperture. The resulting effect is out of focus blur, or *depth of field*. In the last part (Part iii) of the thesis, we target real-time rendering and introduce a new model for depth of field effects in Chapter 5. Defocus blur is commonly seen in everyday life, observable both with our bare eyes and cameras. We investigate the optical mechanism behind this phenomenon and observe that certain effects due to partial occlusion that can be achieved natively with ray tracing are absent in most rasterization based real-time solutions. These effects include blurry image distortions and sharpening effects inside fallout regions of blurred objects that are closer to camera than the focal plane. We propose the *effective circle of confusion*, a screen-space technique that takes partial object obstructions into account and models the optics behind the phenomenon. Thanks to the efficient design of data structures and filtering, we are able to achieve similar effects to ray tracing at a full-HD resolution in less than $2ms$, compared to almost $100ms$ for a state of the art ray-traced implementation.

Each chapter in this dissertation provides a short introduction, a review of prior work and a brief conclusion. Benefits and limitations of methods are presented and the contribution highlighted. We conclude this thesis in Chapter 6 with a review of the presented methods. Finally, an outlook is provided on how the work in this thesis could benefit future work.

1.0.2 *Contributions*

Chapter 2 through Chapter 5 are based on works that are published or under submission during this PhD project. A detailed list is available in Section 6.

Chapter 2 is based on *Primary Sample Space Path Guiding*, procedings of Eurographics Symposium on Rendering - EI & I (July 2018), in collaboration with Dr. Pablo Bauszat, Dr. Jacco Bikker and Prof. Elmar Eisemann.

Chapter 3 is based on *Next Event Estimation++: Visibility Mapping for Efficient Light Transport Simulation*, Computer Graphics Forum, Volume 39, Number 7 (2020). The work is a collaborative result with Prof. Martin Eisemann and Prof. Elmar Eisemann.

Chapter 4 is based on *Geometric Sample Reweighting for Monte Carlo Integration*, Computer Graphics Forum, Volume 40, Number 7 (2021). The work is carried out under supervision of Prof. Elmar Eisemann.

Chapter 5 *Effective CoC: Extended Lens Model for Perspective Cameras in Rendering* is our on-going work currently in submission phase. This is a collaborative work with Dr. Markus Billeter and Prof. Elmar Eisemann.

Apart from the research work in aforementioned projects, the auther is also the main programmer in Chapter 2 through Chapter 4 and co-programmer in Chapter 5 with Dr. Billeter.

Part I

SAMPLING

2

PRIMARY SAMPLE SPACE PATH GUIDING

Abstract

Guiding path tracing in light transport simulation has been one of the practical choices for variance reduction in production rendering. For this purpose, typically structures in the spatial-directional domain are built. We present a novel scheme for unbiased path guiding. Different from existing methods, we work in primary sample space. We collect records of primary samples as well as the luminance that the resulting path contributes and build a multiple dimensional structure, from which we derive random numbers that are fed into the path tracer. This scheme is executed completely outside the rendering kernel. We demonstrate that this method is practical and efficient. We manage to reduce variance and zero radiance paths by only working in the primary sample space.

2.1 INTRODUCTION

Simulating light transport for global illumination effects has been a standard approach for offline rendering. The ultimate goal is to capture both *direct illumination* and *indirect illumination*. Light transport is simulated in such a way that a path, possibly composed of multiple scattering events, connects a light source and the sensor. Direct and indirect illumination are both collected along the path and finally contribute to a pixel, possibly with non-zero luminance. This process is modeled by the *rendering equation* [7]:

$$L(\omega_o \leftarrow \mathrm{x}) = L_e(\omega_o \leftarrow \mathrm{x}) + \int_\Omega L(\mathrm{x} \leftarrow \omega_i) f_s(\omega_o \leftarrow \mathrm{x} \leftarrow \omega_i) G(\mathrm{x}, \omega_i) d\omega_i. \tag{2.1}$$

As the integral of luminance in the rendering equation is high dimensional and generally impossible to solve analytically, *Monte Carlo* methods are usually employed.

Despite the many advantages of Monte Carlo methods, the noise and the slow convergence rate are two major pitfalls. Various *importance sampling* techniques have been developed to reduce variance and accelerate convergence. The common goal is to find an "optimal" distribution that can approximate the true integral. In practice, a distribution that resembles the optimal distribution is known to have good performance [28].
Two of the many open problems addressed in this chapter are:

- too many samples are wasted on paths that do not contribute to the final result;
- inappropriate probabilities are assigned to paths that do not match their actual contribution.

One approach to solve the above problems is to model light paths as *states* in *Markov chain*s (MC) and the rendering problem is transformed into a *Markov chain Monte Carlo* (MCMC) process. This approach is also called the *Metropolis light transport* (MLT) [29]. The core idea of MLT is to mutate existing paths by either altering them a little bit (small mutation) or by completely sampling a new path (large mutation). The major advantage of MLT is that once a path with significant luminance is found, the corresponding region in the path space will be explored extensively, making MLT very suitable for scenes with difficult lighting configurations. The scheme of doing both small and large mutations also guarantees ergodicity. However, MLT suffers from many disadvantages, including start-up bias, poor stratification, and unstable convergence. This prevents its adoption in production especially for animation [78][83].

One new trend is to build distributions based on the results of a few pilot runs and use these distributions for rendering. Most *path guiding* methods use this technique. Existing path guiding schemes usually work in the spatial domain. One intuitive approach is to cache a sparsely distributed set of records in the scene. To make use of them later on, firstly new entries are queried in the spatial structure and then queried results are interpolated to give a final record.

We propose to perform path guiding in a different domain: the *primary sample space* (PSS), i.e., the space of random numbers that are used to generate paths. Our scheme works in two phases: a collecting phase comprising of one or more iterations and a rendering phase. During the collecting phase, we take as input a the random numbers used by a few generated paths together with corresponding luminance values returned by these paths and build a structure that will encode these combinations. During the rendering phase, we sample the previously built structure and use the resulting samples to feed the path tracer. By working in the primary sample space, we are able to build distributions and sample from these distributions outside the path tracing kernel. This makes it easy to add our method to existing solutions while still enjoying the many benefits of being an unbiased Monte Carlo method.

2.2 RELATED WORK

IMPORTANCE SAMPLING Variance reduction is the goal for almost all Monte Carlo optimization methods. For a simplified version of Equation 2.1: $L = L_e + \int L(\omega) f_s(\omega) G(\omega) d\omega$, the corresponding Monte Carlo estimator can be denoted as

$$\hat{L} = L_e + \frac{1}{N} \sum_{i=1}^{N} \frac{L(\omega_i) f_s(\omega_i) G}{p(\omega_i)}, \tag{2.2}$$

where L denotes luminance, f_s denotes the *bi-directional scattering distribution fuction* (BSDF), G denotes the geometry term and p is the probability associated with the current sample ω_i.

The goal of importance sampling is to make the shape of $p(\omega)$ as close to the numerator as possible [47]. The original integrand is composed of three terms: L, f_s and G, one of which in practice is considered the dominant factor. Sampling according to each term has their own advantages. For instance, sampling G favors light coming from above over light coming from the horizon, which complies with the fact that irradiance is cosine weighted. In practice, this cosine weighted sampling is used as the core method to generate an incoming direction for many BSDF implementations, as for most of them there is no easy analytic inversion [85]. Highly glossy materials generally benefit from sampling f_s, as the reflection cone is usually concentrated around a certain direction. Sampling according to L usually works better for

diffuse surfaces since the BSDF is not peaky but it requires global knowledge, which is typically difficult for scenes with unstructured lights and complex visibility.

Multiple importance sampling (MIS) is a technique that combines different sampling strategies [19]. This is done by evaluating the current configuration by using each strategy and weighing the final contribution by balancing all involved pdfs using a certain heuristic. The advantage of MIS is that it automatically gives a distribution that uses all involved strategies, thus making it less likely to yield inappropriate probabilities. One direct application is in *bi-directional path tracing* (BDPT) [20], where for each sample two pre-traced sub-paths are connected vertex-wise. The core efficiency therein is the massive reuse of existing path vertices and MIS is the tool that makes it practical [71].

Many works try to importance sample one or more terms. Few early BSDF models such as *Phong* can be explicitly inverted and consequently be directly importance sampled [12, 14, 25]. Sampling unstructured light sources, known as *next event estimation* (NEE), is common practice[47]. Similarly, sampling structured light sources (such as an environment map) has been investigated in [54]. All of the above methods treat one term in the equation as the dominant factor and provide good results in respective applicable scenarios. Sampling according to the product of multiple terms is studied in [44], where a structured environment map is pre-convolved with BSDFs using wavelets.

PRIMARY SAMPLE SPACE In the original MLT, mutations are done in the path space, a space $\mathcal{P}$ of paths with all possible lengths:

$$\mathcal{P} = \bigcup_{i=2}^{\infty} \mathcal{P}_i, \tag{2.3}$$

$$\mathcal{P}_i = \{\bar{x}_i | \bar{x}_i = \{x_0, x_1, x_2, x_3, \ldots, x_i\}\}, \tag{2.4}$$

where x_i is a path vertex in the spatial domain. However, mutating in path space is difficult: a slight mutation of one vertex could lead to a different path, possibly with different lengths and visibility configurations. Although several methods such as *Manifold exploration* [72] solve the path mutation problems for specular surfaces, in practice mutating in path space is still difficult.

Primary Sample Space Metropolis Light Transport (PSSMLT) offers a simplified version of MLT, where mutations are done by changing random numbers in PSS instead of transforming vertices in the original path space [36]. The high dimensional vector of random numbers is then transformed into actual light paths for final transport simulation. Many MLT variants are based on PSSMLT. *Replica Exchange Light Transport* [61] introduces jumping between mutation strategies. *Multiplexed Metropolis Light Transport* (MMLT) [76] improves efficiency by fixing path length when mutating samples.

The core advantage of PSSMLT is that it transforms an ultra high dimensional problem in path space into a reasonably high-dimensional problem in PSS. This simplification makes MLT simpler to implement and offers another advantage, which is not by design: by altering random numbers only, we are still able to perform local importance sampling at certain locations. Our method is based on this foundation and works in the same space.

PATH GUIDING Since we are simulating light transport, applying importance sampling according to a certain distribution at a scattering event to get an incoming direction translates to guiding a random walk in path tracing.

The *Photon map* was first used to guide path tracing [18], where at each bounce a 2D histogram over the directional domain of surrounding flux information is used to sample an incoming direction. Density estimation of photons is used to approximate the L term. This achieves faster convergence than regular path tracing at the expense of huge memory consumption and slow preprocessing. Instead of using a fixed-resolution histogram, one can also use cone-shaped splats on the hemisphere to represent directional distributions [35]. The latter gives better results in terms of variance reduction than a fixed-resolution histogram at the expense of significantly more computation. Both above methods are particle-based and require caching of certain information, which can be expensive.

A *Gaussian mixture model* (GMM) is used in [78, 83] to represent incoming luminance. Their method features an online learning scheme and the introduction of GMMs gives superior performance over histogram or cone-shaped particle footprints. However, for scenes with small light sources or complex visibility, GMMs tend to be unstable and perform unpredictably. Light transport is modeled as a *Q-learning* process in [89] and previous samples are used to guide successive samples in an iterative manner. This reinforced light-transport scheme, using results from all samples, is thus biased but consistent [91]. Recently a simple path guiding scheme is introduced in [91], where an adaptive quad tree is used to represent a directional domain, while each record is associated with a leaf node in a k-d tree of the spatial domain. This method benefits greatly from averaging incoming luminance over the entire leaf node, which guarantees an abundance of information gathered in each iteration. This method also features a reinforced iterative scheme, with which samples in each iteration are solely guided by the distribution built in the last iteration. This not only guarantees unbiasedness but also reinforces the distribution with minimum effort.

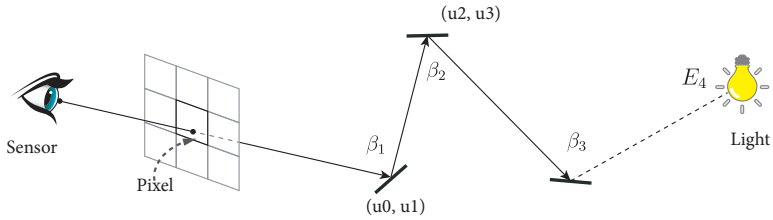


Figure 2.1: A path of length 4, where the light source for the last bounce is connected by either NEE or BSDF sampling. Suppose all three bounces hit non-emissive surfaces. Then the final luminance brought back by this path is $L = \beta_1\beta_2\beta_3 E_4$. In this example, each bounce takes two random numbers to generate an incoming direction. For the first two bounces, the corresponding path record is $\{\{u_0, u_1, u_2, u_3\}, L\}$.

2.2.1 *Distinction*

In previous methods for path guiding, records of directional representation are stored sparsely in the spatial domain. The main distinctive feature of our method is that we work in the primary sample space and we do not explicitly build any spatial-directional structure.

2.3 METHOD

2.3.1 *Formulation*

Equation 2.1 can be re-formulated as a series:

$$L = \beta_0 E_1 + \beta_0\beta_1 E_2 + \ldots + \prod_{i=0}^{n-1} \beta_i E_n + \ldots \tag{2.5}$$

where β_i is the unit throughput $f_s G/p$ at the i-th bounce. E_i denotes the emission of the i-th intersected surface. Note that in this formulation, direct illumination sampling is considered another path.

With reference to Figure 2.1, where a path of length 4 is illustrated, we can model rendering as a function of random number inputs, i.e.:

$$\begin{aligned} L &= f_{\text{path}}(\bar{\mathrm{x}}) \\ \bar{\mathrm{x}} &= T(\{u\}^d) \\ \{u\}^d &= \{u_0, u_1, u_2, u_3, \ldots\}, \end{aligned} \tag{2.6}$$

where $T(\{u\}^d)$ is a function that transforms certain dimensions of random numbers $\{u\}^d$ into a path $\bar{\mathrm{x}}$. By building a multi-dimensional distribution over this space, we approximate the full integral implicitly, i.e., Lf_sG.

It is worth pointing out that PSS is still constrained by the dimensionality. That is, we can only process sample vectors with a limited length. In practice this makes sense, as shorter paths are more likely to bring back luminance than longer ones [8]. To keep the method unbiased, when extracting the random numbers of a path from this structure, we do not stop the path on the position defined by the last random numbers, but continue tracing the path in a Monte-Carlo fashion.

With this formulation, we build a structure with input gathered from a path tracer $\{\{u\}^d, L\}$ so that the distribution contained in our structure is proportional to $\prod_{i=0}^{n-1} \beta_i E_n$. ($d = n \times m$ where n is the bounces we work on, and m is the amount of random numbers used per bounce). In practice, in our examples, we have $n = 2$ and $m = 2$.

2.3.2 *Overview*

An overview of the general process is given in Figure 2.2. In a regular path-tracing solution, *sampler*s are used to generate uniformly-distributed random numbers. The main role of samplers is to maintain a certain level of stratification across dimensions, which is crucial for exploration. In our method, we will use a structure to influence the random number generation. With proper and careful transformation of the sample distribution and probability, we can distribute samples in each dimension such that they can better contribute to the final result.

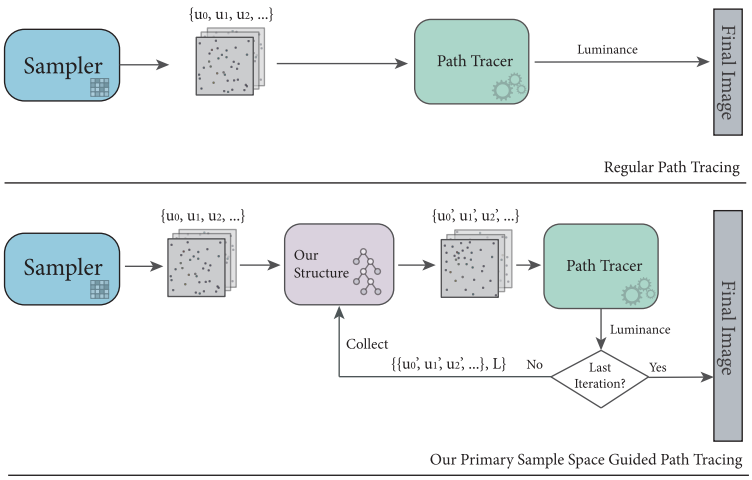


Figure 2.2: A flow chart view of regular path tracing (above) and our method (below). Notice how our method is inserted into an existing pipeline and not inside the kernel.

Our method works in two phases: collecting and rendering, which are iterated. In the first phase, records are collected to build a structure.

In the rendering phase, instead of using uniformly-distributed random numbers to generate paths, we sample the structure to subsequently generate paths. Following the scheme as in [91], we fuse the collecting and rendering within one single rendering budget, i.e., total samples.

More specifically, in the collecting phase, we use all path records from the rendering stage (u, L), where u is the random number vector, which corresponds to the path, and L is the brought-back luminance, add them to the structure and will refine it accordingly. Initially, in the first iteration, the structure is empty and will simply return uniformly distributed random numbers.

During the rendering phase, the successively sampled random numbers will be based on the structure, which is influenced by the path records collected in the previous iteration. This process can be seen as working in the primary sample space (instead of the spatial domain plus the directional domain). Hereby, we are automatically freed from the trouble of building sparsely distributed records and the interpolations incurred.

The rest of the rendering process remains largely untouched; only the random numbers that are used to feed the path tracer are altered. Multiple importance sampling is used to combine samples from our structure and regular uniform samples to ensure that samples cover corresponding domains properly without being stuck in local maxima.

Within one iteration, all samples are generated from the same unbiased distribution, thus results from one iteration remain unbiased. However, when we combine results across iterations, they would be no longer unbiased [89]. Thus, we adopt the scheme as in [91] and only use the last iteration when outputting results to the final image. This process is demonstrated in 1.

Algorithm 1 Pseudo code for PSS path guiding. The only add-ons to a PBRT style render kernel are highlighted in red.

```
1:  procedure RENDER(scene, camera)
2:      ...
3:      for pixel in tile do
4:          kdTree.Init()
5:          for each iteration do
6:              for each sample in iteration do
7:                  ray ← camera.generate(pixel)
8:                  {u}^d ← uniform random numbers
9:                  {u'}^d ← kdTree.SampleTree({u}^d)
10:                 L ← Li(scene, ray, {u'}^d)
11:                 kdTree.Collect({{u'}^d, L})
12:             kdTree.Update()
13:     ...
```

2.3.3 *k-d tree*

In this part, we will detail our structure and how to use it to influence the random-number generation process.

Traditional path guiding methods work in a $5D$ domain where 3 dimensions are used for spatial domain and the remaining for directional domain. In our case, the more bounces we take into account, the higher the dimension. As a common practice in implementation, each scattering event usually takes a fixed amount of random numbers [85]. Recent methods from Metropolis Light Transport also assume fixed dimensions of random numbers when inverting paths to primary samples [92]. It is therefore reasonable to assume that each bounce consumes the same amount of random numbers. We will denote n the amount of bounces and m the involved random numbers, thus our k-d tree spans $n \times m$ dimensions.

For the high dimensional problem that we are facing, one proper structure is the *multiple dimensional binary tree* [1], also known as the *k-d tree*. The advantage of using a k-d tree instead of a hyper grid or any other structure is that k-d trees have $O(N)$ memory efficiency and $O(\log N)$ query efficiency. We use a k-d tree that switches axis in a round robin fashion and always splits in the center of current node range.

2.3.3.1 *Data Structure Design*

We use a pointer-less indexed tree structure, so the left child is always next to the parent in memory and a child pointer indicates the right child. We use a compact node structure of only 8 bytes, the child offset for inner nodes, which is used as a sample counter in leafs, is encoded on 31 bits, a leaf flag one bit, and the remaining bits are used to encode a node value.

2.3.4 *Tree Construction*

Our method follows in spirit three dimensional binary trees in the spatial domain [91]. Whenever a path is complete, all path vertices are traversed and luminance from a vertex onward is accumulated to the leaf node where the vertex is located. Within the spatial tree leaf, all accumulated information is averaged and added to an associated directional quad-tree in world space. Similarly, we use all path records within a tile of pixels to construct our sample-space tree. We also adopt an iterative scheme, where samples from the current iteration are sampling results from the tree built in the previous iteration. Using samples from the same distribution defined by this tree gives unbiased results.

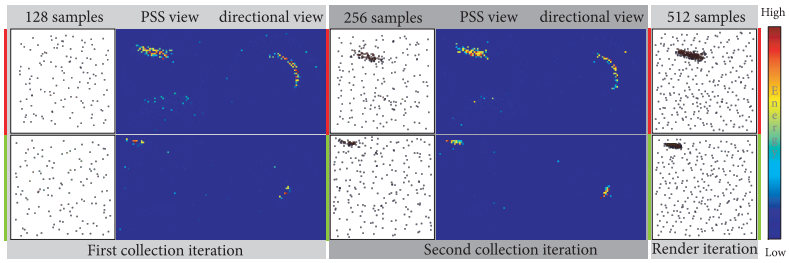


Figure 2.3: Visualizations of the first two dimensions in k-d trees in two regions. In this case we use a two dimensional tree. Notice in the first iteration, samples are uniformly distributed. In the second iteration and the third iteration, samples are warped. Other regions remain covered properly thanks to MIS. Notice also that the directional views are merely for visualization, all collection and sampling are done in PSS.

For the initial iteration, with an empty tree, the output remains uniform. Figure 2.3 shows a visualization of two selected trees across iterations.

2.3.5 *Tree Sampling*

We maintain two trees: one for collecting and the other for sampling. After each iteration, we copy the collection tree to the sampling tree. We descend the k-d tree as in [26] from the root node and select a child according to the values contained therein compared to a random number. In this way, we change the sampling process to follow previously recorded contributing samples.

Specifically, for node i, with a value recorded as v_i, children nodes j and k, the naïve probability of selecting either one is set to:

$$p_j{}' = \frac{v_j}{v_i} = \frac{v_j}{v_j + v_k} \tag{2.7}$$

$$p_k{}' = \frac{v_k}{v_i} = \frac{v_k}{v_j + v_k}. \tag{2.8}$$

Since we are dividing node i in half, the correct probability should include this scaling factor. Thus the correct probability is;

$$p_j = \frac{p_j{}'}{1/2} = \frac{2v_j}{v_j + v_k} \tag{2.9}$$

$$p_k = \frac{p_k{}'}{1/2} = \frac{2v_k}{v_j + v_k}. \tag{2.10}$$

The denominator in the expression of the current depth will cancel all the way to the root node and the probability of a node i being sampled will depend on node depth and node value:

$$p_i = \frac{2^{d_i} v_i}{v_{\text{root}}}, \tag{2.11}$$

where d_i is the depth of node i.

Once at the leaf node, the leaf node represents a $n \times m$-dimensional range (the range covered by the k-d tree leaf). We then sample a uniformly distributed vector $u \in [0,1)^{n \times m}$ and multiply it with the ranges to derive our new random variables to be used to define a path. To support paths of arbitrary lengths, beyond n bounces, we can simply feed the path tracing kernel with uniform random numbers after having used the components of u.

To be able to weigh the contribution of the chosen path correctly according to its probability, we keep track of the probability of the ranges that are represented in the leaf in form of a vector that follows the pattern of $(pdf_0^0, \ldots, pdf_{m-1}^0, \ldots pdf_{m-1}^{n-1})$. If all values in the tree are one, these probabilities would correspond to the size of the k-d tree cell. With the values in the nodes, these probabilities need to be adapted accordingly during the descent.

2.3.6 *Rendering*

Having derived random numbers to produce a path, we need to properly weigh its contribution. In our case, for bounce n, the sample probability p_{sample}^n is comprised of two parts; the conditional probabilities of previous dimensions and the probabilities of the current bounce:

$$p_{\text{sample}}^n = p(p^n | p_{\text{sample}}^{n-1}) = \prod_{i=0}^{n-1} p_{\text{sample}}^i \prod_{j=0}^{m-1} pdf_j^n \tag{2.12}$$

where p_{sample}^i is the sample probability of bounce i, p_j is the probability of the j-th random number for bounce n. This sample probability is multiplied with the Jacobian $||J||$ of the BSDF sampling method, e.g., for cosine weighted sampling where the sample is transformed into a direction ω:

$$p(\omega) = \frac{1}{||J||} \times p_{\text{sample}} = \frac{\cos\theta}{\pi} \times p_{\text{sample}}. \tag{2.13}$$

To avoid over exploration of certain region in PSS due to our reinforced scheme, MIS is used to combine samples from our structure and the original uniform samples. A guide probability is used to balance between the two. In all our tests, we use a guide probability of 0.5.

2.3.6.1 *Record Collection*

We start with an empty tree that has a uniform output. After the first iteration, to insert samples from the rendering stage, we take a path with random numbers u and luminance L. We traverse the tree according to u. At inner nodes, we simply add the path record value and then check into which child to descend. At leaf nodes, we not only add a path record value, but also increase the count variable in the corresponding leaf. When a threshold for the count is reached and increasing depth is still permissible, we split the current leaf node into two and update index information accordingly.

2.4 IMPLEMENTATION AND RESULTS

2.4.1 *Implementation*

We implemented our method in PBRT [85] as a guided unidirectional path tracing integrator. The k-d tree is implemented as an acceleration structure. We build a k-d tree for each tile, as this requires minimum parallel implementation effort and distant pixels from other tiles are less likely to share correlated illumination condition.

Memory-wise, due to our scheme of working within tiles, the maximum memory is bounded by the number of threads times maximum memory for one tree. In our test where we use an eight core machine and a k-d tree max depth of 20, memory is in theory bounded by 128 MB. However in our low dimensional tests, we never observe higher memory consumption than 32 MB for all k-d trees during the whole rendering process. Since in our scheme memory is bounded and not an issue, we focus our verifications on other criterias.

2.4.2 *Comparing Standards*

In all our tests, we compare our results with non-guided results. Since the main objective of this chapter is to explore the possibility of guiding paths in PSS, we do not compare our results with existing path-guiding methods that work in the spatial-directional space.

Two comparing criteria corresponding to the two problems mentioned in Section 2.1 are:

- **Zero-radiance path ratio** refers to the ratio of full paths that do not contribute to final results, as well as the NEE sampled direct-illumination subpaths that are occluded.
- **Mean square error** (MSE) is a common tool to evaluate convergence rate. Typically, the lower the MSE, the lower the variance.

To better visualize the difference between our method and standard path tracing, we off NEE at the first bounce. We leave NEE on from the second bounce onwards to improve finding non-zero light paths.

We test our method in four closed scenes with complex illumination configurations:

- **Ajar door scene**: most of the image is lit by indirect illumination
- **Dinning room scene**: featuring complex visibility because of the window blinds
- **Staircase scene**: most of the image is lit by an area ceiling light, regions under the staircase have difficult light configuration
- **Kitchen scene**: complexity visibility

All scenes are rendered at a resolution of 800×450 and a tile size of 16×16.

2.4.3 *Results*

We render four scenes with different sample budget (collection plus rendering) and compare them against a regular path tracer with the same sample budget. The reference images are rendered with the regular path tracing integrator using around $10k$ samples per pixel. In all our guided renderings, we use $1/8$ total samples for the collection phase and the remaining $7/8$ samples for the rendering phase. For sample rate above 1024, we use two iterations for the collecting phase, each using an equal amount of samples.

Visual results are shown in Figure 2.4, Figure 2.5, Figure 2.6 and Figure 2.7. In all our figures, we use the notation of "collection" samples $+$ "rendering" samples, e.g, $64 + 448$ means 64 samples are collection samples and the remaining 448 are rendering samples. Convergence rates (MSE-spp) of all tests are plotted in Figure 2.8. Zero radiance paths statistics are given in Table 2.1.

OBSERVATIONS Comparing our guided results with non-guided results at the same budget, we have the following observations:

- In all our tests, our guided results show better overall convergence in terms of MSE, where in the best case we managed to reduce costs by 39.5% in Figure 2.7.

- In regions where the first bounces make a major difference, our results show obvious improvement over non-guided results. See, e.g., red region in the *dining room* scene in Figure 2.5. In regions, where more than two bounces are needed to connect the light

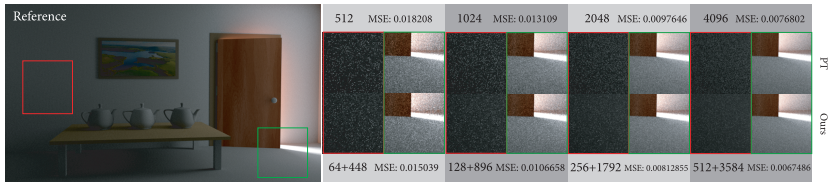


Figure 2.4: The *Ajar* scene.

source to the sensor, the improvement is minor or none.

- In all our tests, we managed to reduce the ratio of non-contributing paths. In the best case by 12.77% in the *Ajar* scene.

- With our iterated scheme, this ratio drops further, as the sample budget increases. While for the non-guided tests, this ratio stays at the same level. This is to be expected, since samples in later iterations are guided towards lit regions.

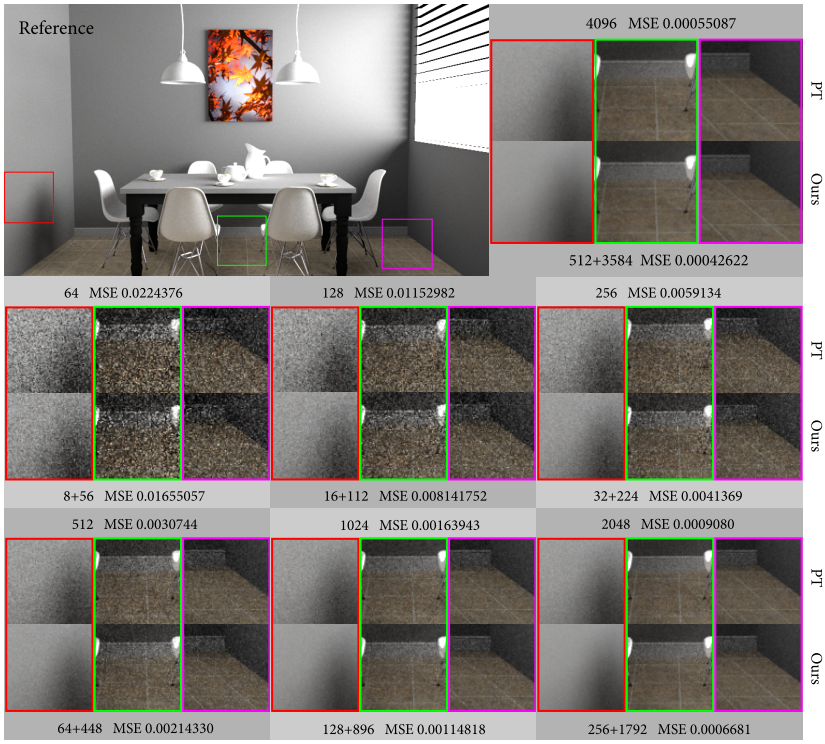


Figure 2.5: The *Dining room* scene.

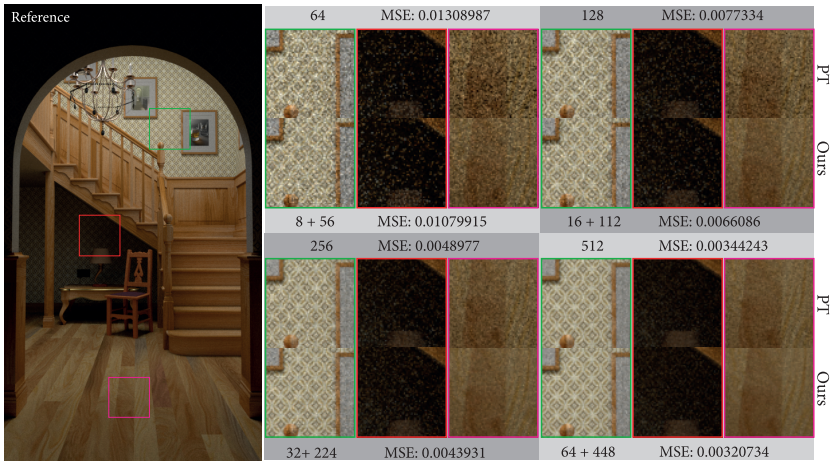


Figure 2.6: The *staircase* scene. Notice the improvement in pink regions.



Figure 2.7: The *kitchen* scene.

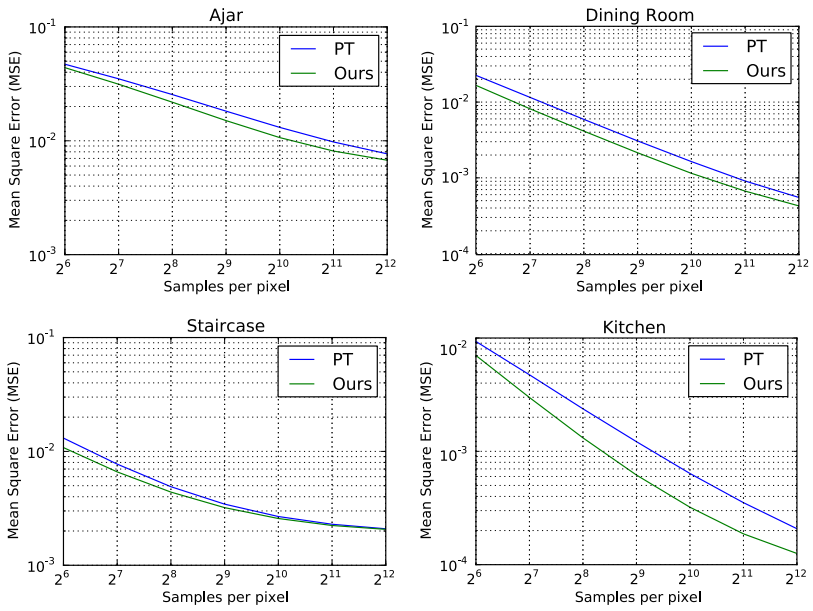


Figure 2.8: Convergence plot of all our tests.

Scene	**Method**	64 SPP	128 SPP	256 SPP	512 SPP	1024 SPP	2048 SPP	4096 SPP
Ajar	PT	96.58%	96.58%	96.59%	96.59%	96.58%	96.58%	96.59%
Ajar	Ours	94.70%	93.68%	92.37%	90.72%	88.71%	86.50%	84.26%
Dining	PT	87.13%	87.13%	87.13%	87.13%	87.13%	87.14%	87.14%
Dining	Ours	87.33%	87.24%	87.08%	86.92%	86.75%	86.62%	86.53%
Kitchen	PT	97.50%	97.50%	97.50%	97.50%	97.50%	97.50%	97.50%
Kitchen	Ours	96.46%	95.52%	94.20%	92.61%	90.94%	89.40%	88.11%
Staircase	PT	66.51%	66.52%	66.52%	66.52%	66.52%	66.52%	66.52%
Staircase	Ours	65.93%	65.38%	64.61%	63.72%	62.74%	61.76%	60.76%

Table 2.1: Zero radiance path ratios (numbers are in %) of all our tests. Notice in all our tests, our method manages to bring down zero radiance paths ratio. Notice also that as more and more samples are used for the collection phase, zero-radiance path rates drops accordingly. While in the non-guided tests, this ratio remains at the same level regardless of the sample rate. This demonstrates the effectiveness of our scheme.

2.5 DISCUSSION

Our method reduces non-contributing paths and overall variance, showing the effectiveness of the primary sample space-path guiding. When comparing same sample rate quality, especially in the *dinning room* scene and the *kitchen* scene, our method significantly outperforms unguided results.

In all our tests with building distributions in primary sample space, results show that this approach is not only viable, but also efficient. Within a fixed rendering budget, even 1/8-th of the samples will result in a proper distribution that improves the result. It may not work in all cases, but the bottom line is that it does not introduce variance. One of the core advantages of our approach is that we are working completely outside the rendering kernel, all we need to do is modify samplers so that their outputs are no longer uniform.

HIGHER DIMENSIONS In our tests, we are only working with two bounces and use two random numbers for each bounce (hence, $n = m = 2$). The result we get in the *staircase* scene shows that for regions that can reach the light source via a two-bounce path, the improvement is evident. However, for regions that require more bounces, our method provides little help for them. These regions remain noisy, similar to the result of an un-guided path tracer.

ADAPTIVE STRUCTURES In our implementation, we use one k-d tree per tile and average information gathered in this tile. It can be helpful to make our structure adaptive. The fact that most of the obviously improved regions in our tests are flat and smooth surfaces also complies with this point. To this extent, we could define tiles according to scene geometry, e.g., the surface normal difference.

COMBINATION WITH EXISTING METHODS The independence of the rendering kernel and the spatial domain, makes it easy to combine our scheme with existing path-guiding methods using directional records in spatial structures. One possible combination is to work in a “relay race" mode, in which the duty of path guiding is handed over to the spatial-directional structure from our primary sample space structure after a few bounces.

2.6 CONCLUSION AND FUTURE WORK

In this chapter, we presented a novel scheme for unbiased path guiding in primary sample space, which we showed to be practical and efficient. Even when sparing only a small portion of the total rendering budget on the collection step the results typically lead to a better outcome than

using the entire budgets for rendering alone. Our scheme of working in primary sample space is easy to implement and can be comfortably added onto any existing Monte Carlo rendering system.

For future work, we would like to extend our idea to higher dimensions and to bi-directional path tracing.

3

NEXT EVENT ESTIMATION++: VISIBILITY MAPPING FOR EFFICIENT LIGHT TRANSPORT SIMULATION

Abstract

Monte-Carlo rendering requires determining the visibility between scene points as the most common and compute intense operation to establish paths between camera and light source. Unfortunately, many tests reveal occlusions and the corresponding paths do not contribute to the final image. In this chapter, we present *next event estimation++ (NEE++)*: a visibility mapping technique to perform visibility tests in a more informed way by caching voxel to voxel visibility probabilities. We show two scenarios: Russian roulette style rejection of visibility tests and direct importance sampling of the visibility. We show applications to next event estimation and light sampling in a uni-directional path tracer, and light-subpath sampling in Bi-Directional Path Tracing. The technique is simple to implement, easy to add to existing rendering systems, and comes at almost no cost, as the required information can be directly extracted from the rendering process itself. It discards up to 80% of visibility tests on average, while reducing variance by ~20% compared to other state-of-the-art light sampling techniques with the same number of samples. It gracefully handles complex scenes with efficiency similar to Metropolis light transport techniques but with a more uniform convergence.

3.1 INTRODUCTION

Unbiased Monte Carlo (MC) rendering algorithms, such as Path Tracing (PT), Bi-Directional Path Tracing (BDPT) and Metropolis Light Transport (MLT) are among the most versatile and robust rendering algorithms today. They use importance sampling techniques to efficiently solve the rendering equation [7] by integrating over light paths between the camera sensor and light emitter. An integral part, and often the most expensive, is building vertex connections between camera and light subpaths. These are solved by tracing rays between vertices to determine visibility. While an acceleration structure (usually a *bounding volume hierarchy*, BVH) can lower the cost of such queries, these tests are numerous and, depending on the outcome, there is no contribution to the final image. In fact, any occluded connection is discarded. The problem is even more aggravated by the fact that the number of camera and light vertices is generally linear in the number of pixels and light sources, whereas the number of visibility tests can be quadratic in the number of path vertices. For example, in a BDPT scenario for subpaths of length 8, internal connections comprise already ~80% of the acceleration structure visits.

We present a novel and versatile visibility mapping technique to importance sample the camera/light path connections. We precompute a voxel-to-voxel visibility within the scene, which is the probability of an unoccluded connection between a random point in the first voxel and a random point within the second voxel. For efficiency, we reuse information gathered during the initial light/camera path generation, leading to a construction overhead in the range of a few hundred milliseconds to a few seconds – a fraction of the total rendering time. We save the visibility information in a matrix-like map where the rows and columns represent different scene voxels and each entry reflects their approximate mutual visibility. During rendering, we can use the structure to estimate the probability of the success of camera-to-light vertex connections before any explicit tests. Our visibility map can be applied in various ways, e.g., Russian roulette style rejection sampling or direct importance sampling.

We name our method *next event estimation*++ (NEE++). Next event estimation (NEE) refers to direct illumination sampling, for which explicit light emitter information is given and sampled to form a potentially connected path segment. With our proposed method, we not only estimate the next event(s), but also provide an estimate of the likelihood that the connection succeeds. Our approach can drastically reduce the number of point-to-point visibility tests by up to 80% and shows an approximate 20% variance reduction compared to other techniques with the same number of samples.

Specifically, our main contributions are:

- A view-independent visibility mapping, encoding voxel-to-voxel visibility, generated at a low additional cost;
- Unbiased estimators built atop of our visibility mapping;
- Several example applications of our technique.

3.2 RELATED WORK

Visibility is a key component of rendering and the topic too broad to cover in this chapter. Consequently, we focus only on publications reducing the amount of tests and improving sampling in the context of solving the rendering equation [7].

In his seminal work Veach [31] already proposed using Russian roulette to reduce visibility tests between camera and light subpaths but without global scene information. This idea was recently extended to stratified sampling in the connection space to avoid sampling connections, which are too similar and contribute little new information to the image [94]. The opposite of early termination has been demonstrated with splitting in [87].

PCBPT by Popov *et al.* [82] and the close followup work by Nabata *et al.* [97] importance sample connections from a given eye vertex to a set of presampled light subpaths. They compute probabilities of connecting to the light vertices for a subset of eye vertices and interpolate these for the rest. Their technique provides a trade-off between accuracy in the probabilities and computation time but, contrary to ours, requires an expensive search query in a kd-tree for each camera vertex and is recomputed iteratively to avoid visible artifacts during convergence. The necessary adjustments for this approach also can make it difficult to integrate into existing pipelines. The information needed for our approach requires only a few simple look-ups per camera vertex to estimate the probabilities, is computed only once (though it could be updated as well) and comes basically with negligible additional cost for creation.

For a somewhat simpler problem of precomputing binary visibility for direct lighting (e.g., a surrounding environment map), computing sparse hemispherical caches that are interpolated during rendering to steer importance sampling of the map has been shown to work well [53]. Augmenting these caches with additional shading information, like depth, normals, material information, they can even serve as an approximate scene representation to accelerate later bounces in unidirectional path tracing[75]. However, the direct usage, especially in earlier bounces, introduces bias, which is also present when clustering visibility into discrete bins and using these directly for visibility testing [74]. Still, visibility tests are significantly reduced and accelerated, a motivation that our work shares. In contrast, our technique is unbiased,

more versatile (as it is applicable to many rendering techniques) and easier to integrate into existing rendering systems.

Caching pre-computed visibility information between spatial subdivisions and arranging such data in memory for a second render pass has been investigated in [16, 17, 21, 23, 34, 56].

Direct illumination is treated in [17] by using a ratio of prior shadow tests as the final contribution discounter for other light samples. Hard thresholding and the direct use of the tests is biased and only addresses direct illumination. Tellier *et al.* introduces scene voxelization to form a binary visibility graph. In format this is similar to our visibility map but binary. The direct use of the binary information in the hierarchy for direct illumination calculation inevitably leads to biased results [16]. Fernandez *et al.* introduces similar idea by sparsely caching visibility status (fully visible, partially visible and fully occluded) for light sources. Shadow rays are only tested for partially visible light sources according to the cache. This introduces bias in the other two cases [34]. Lacewell *et al.* propose to cache such information in BVH nodes, providing an early out criteria for going down the hierarchy. In principle, this re-allocates shadow-ray tests but the employed hard thresholding introduces system bias [56]. These related works are biased and have very specific application, while our way of caching visibility as probability does not suffer from bias and is more general.

As an alternative to influence the sampling process, some work also proposes a two-pass strategy [21]: first a radiosity solution is computed using simplified geometry, which is view independent. The resulting surfaces are considered light sources. Next, a regular grid is produced, in which each grid cell contains the list of visible and strongly-contributing light sources. To test visibility a single point-point test is performed. During rendering, a gathering is performed at each pixel by only testing the lights connected to the cell containing the pixel. The method is biased, handles only Lambertian materials, and applies a coarse visibility sampling. Our method does not store light-sources per cell but more general visibility relations between regions of space. We avoid bias and handle arbitrary materials. Further, the cost of our preprocess is negligible.

A trend in production rendering, quickly established as the state of the art, is to guide the sampling when extending camera or light subpaths. Here, we mention only a few examples and refer the interested reader to [102]. A successful option is to learn optimal directions for ray sampling [77, 91, 95, 98, 99]. Kroes *et al.* [86] process environmental light and visibility jointly to improve direct light estimation at scattering events in volumetric data. Similarly, Herholz *et al.* [101] sample scattering directions according to the product of phase function and incident radiance, which includes visibility and supports multiple scattering. Keller *et al.* utilize visibility information that is encoded

in pre-cached photon maps in section "*Guiding and Shadow Rays*" in [102]. Visibility is reused for illumination sampling and the technique is demonstrated to improve convergence. Our technique could be viewed as a generalization as we target spatial-spatial visibility, regardless of light source or not. This enables the multiple usage as demonstrated in this chapter. Though related, as these techniques reduce the number of required samples, path guiding is essentially orthogonal to our technique. It guides the sampling usually during path creation, whereas ours improves the success rate of shadow tests and could be integrated to improve next event estimation.

3.3 VISIBILITY MAPPING

Our method consists of three steps: visibility-map creation (Section 3.3.1), visibility caching (Section 3.3.2) and visibility use (Section 3.3.3), see Figure 3.1.

The visibility map encodes visibility relations between points in the scene. This information is gathered during the rendering process and a fast preprocess. Once established this visibility information steers the sampling process by probabilistically rejecting visibility tests that have little chance to succeed. This solution can be used for unbiased estimates.

3.3.1 *Data Structure*

The visibility map $\tilde{V} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ is a function that, given two events $x', x'' \in \mathbb{R}^3$, returns the estimated probability of a ray from x' to x'' to not intersect the scene. To render the encoding practical, $\tilde{V}$ uses an implicit uniform voxel grid, englobing the scene of resolution D^3 (Figure 3.1 (0.)) with each voxel having a unique index in $\{0, ...D^3 - 1\}$. For each pair of voxels (V_i, V_j), we have $\tilde{V}(x', x'') = \tilde{V}(y', y'')$ for all events $x', y' \in V_i$ and $x'', y'' \in V_j$. We encode $\tilde{V}$ as a 2D matrix, where each entry stores the mutual visibility estimate between the corresponding voxels given by the row and column indices (Figure 3.1 (1.)). We obtain these visibility estimates as described in the following section.

3.3.2 *Obtaining Visibility Information*

The ratio of unoccluded visibility between events in two voxels can be naively determined by randomly choosing respective points within the voxels and testing their mutual visibility. The averaged visibility converges to the wanted result. The quadratic complexity of this step and inclusion of empty voxels makes it suboptimal. Similarly, voxels

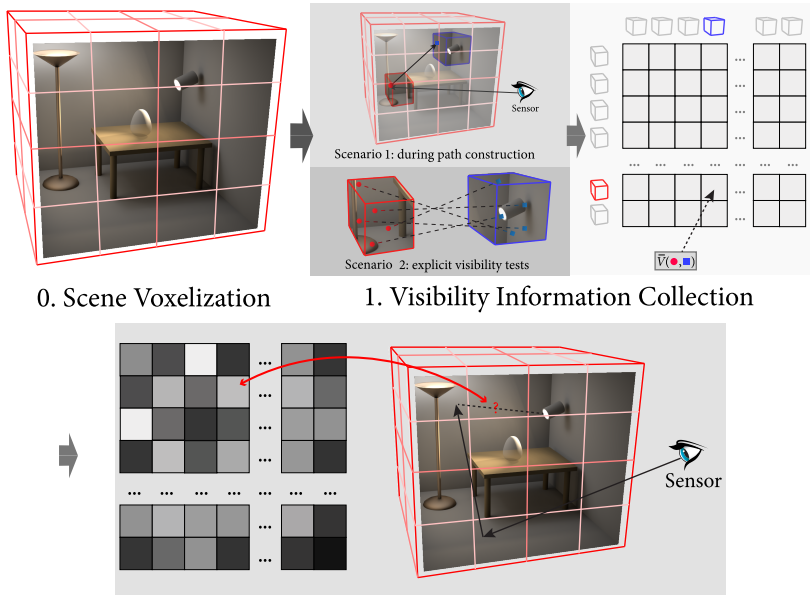


Figure 3.1: NEE++ Overview. Visibility information is obtained during regular path construction and explicit spatial-spatial testing. The highlighted voxel pair in (1.), correspond to an index in the visibility map, where the average visibility is accumulated. As an application (2.), the quality of a shadow ray can be estimated by querying the estimated visibility of the voxel pair containing the path vertices.

within opaque watertight objects are of no relevance but would still be tested.

To determine relevant connections and reduce the overhead, we update $\tilde{V}$ stochastically. Each entry is initialized to an $\epsilon > 0$ (in practice, 10^{-4}), which ensures that no paths are entirely rejected in the later stages to maintain unbiasedness. We then update $\tilde{V}$ during the initial camera and light path creation (including the correspondingly tested shadow rays) of the standard rendering step. We accumulate the successful and unsuccessful visibility tests (Figure 3.1(1.), Scenario 1) and save their ratio in $\tilde{V}$. This step comes at almost no additional costs in a standard rendering pipeline.

We can perform additional visibility tests to refine the result (Figure 3.1(1.), Scenario 2). This second step is especially useful for offline rendering as it improves precision and has negligible cost compared to the overall rendering time. In practice, we use 16 additional tests but restrict them to entries that have been increased from ϵ after the first step.

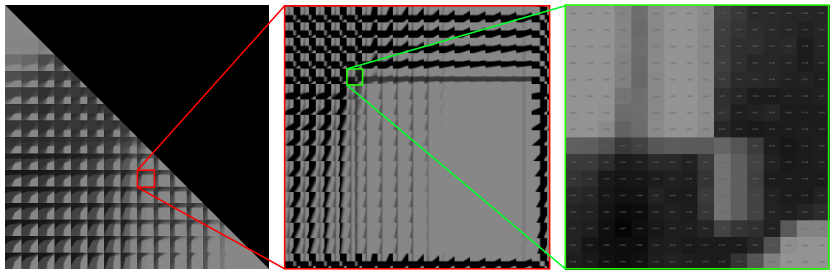


Figure 3.2: A 4096×4096 visibility map for the *Cornell Box*, which uses 32 MB, including closeup images.

Figure 3.2(A) shows a visualization of our visibility map where brighter intensity values represent a higher ratio of successful visibility tests between the respective voxels. We save only the lower-triangular part of the map as a linearized vector due to the inherent symmetry, Figure 3.2(A, left). The visibility map converges as more visibility samples are used. Convergence behaviour appears to be similar to typical MC integration techniques. In our experiments we find that, with 64 samples per entry the result is already close to the reference solution ($1k$ samples). As $\tilde{V}$ can be reused for any viewpoint, it can be applied over several frames of a static scene without updates. This fact and the low amount of necessary rays, makes the creation overhead of the visibility map negligible in most rendering scenarios.

3.3.3 *Using Visibility Information*

Section 3.1 mentioned two possible ways to use our map: rejection sampling or direct sampling. Since we are targeting a full light-transport simulation, using cached visibility to calculate illumination would result in a biased estimator. Instead, we opt for a stochastic use to solve the rendering equation (here, slightly reformulated to keep the visibility function V separate):

$$L(x', x) = L_e(x', x) + \int_A f_s(x'', x', x) L(x'', x') G(\cdot) V(x'', x') dA''(x''), \tag{3.1}$$

where $G(\cdot) = G(x'', x') = \cos\theta_i'' \cos\theta_o' / \|x'' - x'\|^2$. The equation describes the recursive transport of light energy from area A via scattering event x' to event x, as well as the direct energy from event x' to event x. The visibility term $V(x'', x')$ is 1 if x'' and x' are mutually visible, or 0 otherwise.

3.3.3.1 *Rejection Sampling*

Russian roulette early termination is widely used during path construction to randomly terminate paths with low throughput [85]. With respect to our problem, we can apply the same technique to the visibility term. With simplification of notations from Equation 3.1 by leaving out parameters, we can write the estimator as:

$$\hat{L}_{rr} = L_e + \begin{cases} \frac{f_s LGV(x'',x')}{\tilde{V}(x'',x')p(x'')} & \xi < \tilde{V}(x'',x') \\ 0 & else \end{cases}, \xi \sim U[0,1] \tag{3.2}$$

This estimator is unbiased:

$$\begin{aligned} E[\hat{L}_{rr}] &= L_e + \tilde{V}(x'',x') \times E[\frac{f_s LGV(x'',x')}{\tilde{V}(x'',x')p(x'')}] + (1 - \tilde{V}(x'',x')) \times 0 \\ &= L_e + E[f_s LGV] = L. \end{aligned}$$

Russian roulette does *not* reduce variance – actually it is known to increase variance. Nevertheless, by taking early outs, we avoid expensive visibility tests when they are not likely to contribute to the final image. By pruning unnecessary tests, we can allocate resources to constructing additional paths. This is a typical scenario for constant operation time optimization.

3.3.3.2 *Direct Sampling*

Our visibility map can also be used to guide importance sampling. Given a set of M candidates $\{x_i | x_i \in R^3, i = 1, \cdots, M\}$ to connect to a vertex x_0 (e.g., vertex connections in BDPT or light sampling in PT), we can build a 1D discrete distribution $P_v(x')$. We can sample this distribution and get candidate x_j and its associated probability $p_v(x_j) = \tilde{V}(x_j, x') / \sum_i \tilde{V}(x_i, x')$. The NEE++ estimator for Equation 3.1 is then $\hat{L} = L_e + \hat{L}_v$, where:

$$\hat{L}_v = \frac{1}{N} \sum_{i=1}^{N} \frac{f_s LGV(x_i, x')}{p_v(x_i)}. \tag{3.3}$$

Since by design our $p_v(x)$ is guaranteed to satisfy $p_v(x) \geq 0$ and $\sum p_v(x) = 1$, $p_v(x)$ is a valid pdf and our estimator unbiased.

3.4 APPLICATIONS OF VISIBILITY MAPPING

In this section we present three example applications of NEE++. Additional possibilities will be discussed in Section 3.7.

3.4.1 *NEE++ Rejection Sampling Shadow Test*

Following Section 3.3.3.1, we present a direct application of *rejection sampling*. When constructing a backwards-traced path from the sensor to a possible light source in unidirectional path tracing, direct illumination can be explicitly sampled at each scattering event (NEE). For each newly added path vertex, instead of explicitly testing the connection directly, we reject the test based on the probability p_{map} in $\tilde{V}$. For unbiasedness the kept samples need to be weighted with $1/p_{map}$.

The pseudo code of the NEE++ rejection sampling is given in 2. The difference of our estimator from a standard path tracer lies in lines 4–6: stochastic visibility test with probability p_{map} and light-sample weighting.

Algorithm 2 Pseudo code of NEE++ Rejection Sampling Shadow Test in unidirectional path tracing

```
1: procedure LightSampling(scene, path, visMap)
2:     L' ← 0
3:     [sample_light, p_light] ←scene.sampleEmitter()
4:     p_map ←visMap.query(path.last, sample_light)
5:     if ξ ≤ p_map then
6:         L' ← f_s LGV(path.last, sample_light) / (p_light p_map)
7:     return L'
```

3.4.2 *NEE++ Direct Light Sampling Strategy*

Using $\tilde{V}$ we can importance sample the visibility term in the rendering equation by adapting light sampling. LightSampling uses a prebuilt index list of voxels containing light emitters (3, line 1) which is computed before rendering the image. To generate a light sample for a given path vertex x, we first create a 1D distribution of visibility from x to the light-containing voxels using $\tilde{V}$ (3, line 4–5). Next, we draw a sample from this distribution to choose a voxel (3, line 6), from which the light sample is drawn (3, line 7). In the degenerate case, the 1D distribution is constant, we obtain a uniform light sampling.

3.4.3 *NEE++ Bi-Directional Subpath Sampling*

Our visibility mapping can also be applied to BDPT by using NEE++ as a sampling strategy for choosing light subpaths out of a set of sampled

Algorithm 3 Pseudo code of NEE++ as a Direct Light Sampling Strategy in unidirectional path tracing

```
1: lightIDs←BuildLightIndex(scene)
2: procedure LightSampling(scene, path, visMap)
3:     lightDistro← 0
4:     for i in range (lightToVoxels.size()) do
5:         lightDistro[i]+=visMap.query(path.last, lightIDs[i])
6:     voxID ← Sample1D(lightDistro, ζ)
7:     [sample_light, p_light] ← scene.sampleEmitterInVoxel(voxID)
8:     return f_s LGV(path.last, sample_light) / p_light
```

light subpaths for a given camera subpaths (Figure 3.3). We reformulate the rendering equation in *path space* as:

$$L = \int_{\mathbb{X}} \int_{\mathbb{Y}} \textsc{Connect}(\bar{X}, \bar{Y}) d\bar{X} d\bar{Y}, \tag{3.4}$$

where the symbols are respectively:

- $\mathbb{X}$: the space of all camera subpaths;
- $\mathbb{Y}$: the space of all light subpaths;
- $\bar{X}$: a differential camera subpath in space $\mathbb{X}$;
- $\bar{Y}$: a differential light subpath in space $\mathbb{Y}$;
- $\textsc{Connect}(\cdot)$: weighted bi-directional subpath connection that connects two subpaths with multiple importance sampling.

The full path space is the bi-directional connection of two subpath-spaces, where all camera subpaths are connected to all light subpaths. As a condition, we assume that camera and light subpaths are generated independently from each other, as is the case for standard BDPT.

For integration of Equation 3.4, we derive the following MC estimator:

$$\hat{L}^{BiDir} = \frac{1}{M} \frac{1}{N} \sum_{i=1}^{M} \sum_{j=1}^{N} \bar{X}_i \bar{\circledast} \bar{Y}_j. \tag{3.5}$$

M and N denote the number of samples for camera and light subpaths respectively. As $\hat{L}^{BiDir}$ is an unbiased estimator, we can think of the solution of $\hat{L}^{BiDir}$ as a progressive solution that can be iterated and accumulated on demand until convergence. Consequently, the set of light paths $\bar{\mathbf{Y}}$ in each iteration can be generated beforehand for each camera sample $\hat{X}_i$.

Instead of connecting $\bar{X}_i$ to all subpaths in the set $\bar{\mathbf{Y}}$, we can rewrite Equation 3.5 as an importance-sampling problem, where only one light subpath is selected with a certain probability for each camera subpath:

$$\hat{L}_v^{BiDir} = \frac{1}{M}\sum_{i=1}^{M}\frac{\bar{X}_i \bar{\circledast} \bar{Y}_j}{p_v^{BiDir}(\bar{Y}_j|\bar{X}_i))}, \tag{3.6}$$

where $\bar{Y}_j$ is the sample drawn from $\bar{\mathbf{Y}}$ with probability $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$.

Here $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$ is a valid probability mass function, a discrete probability distribution (the set of light subpaths $\bar{\mathbf{Y}}$, from which to sample $\bar{Y}_j$), rather than a continuous pdf.

We define the value of $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$ as the normalized sum of visibility values in $\tilde{V}$ to connect all vertices in $\bar{Y}_j$ to $\bar{X}_i$ to increase the likelihood for vertices to connect. $\hat{L}_v^{BiDir}$ is still an unbiased estimator.

To obtain $\bar{X}_i$, $\bar{Y}_j$ and p_v^{BiDir}, we generate one camera subpath at a time and N light subpaths that get reused for all camera samples of one iteration and construct p_v^{BiDir} from this. The step by step pseudocode for BDPT with NEE++ and building p_v^{BiDir} is given in 4 and described in the following.

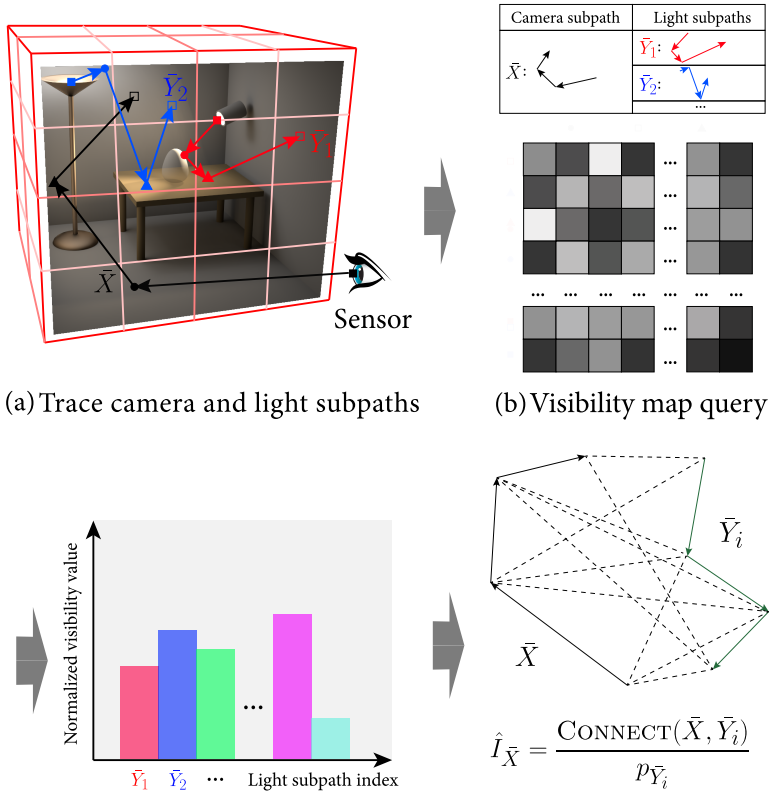
In each iteration, we create a subset $\bar{\mathbf{Y}}$ of light subpaths. For each camera subpath $\bar{X}_i$, we build p_v^{BiDir} (4, line 5) to select a corresponding light subpath $\bar{Y}_j$ to connect to (4, line 6). To build p_v^{BiDir} (4, line 9–13), we estimate a probability for $\bar{X}_i$ to connect successfully to each of the light subpaths $\bar{Y}_j \in \mathbb{Y}$. The probability for selecting a light subpath $\bar{Y}_j$ is the normalized accumulated visibility to connect all vertices of $\bar{X}_i$ to $\bar{Y}_j$ (4, line 14–18).

3.5 RESULTS

All implementations and experiments are carried out within the framework of PBRT [85] on an i7 6800K 3.4GHz six core PC with 32GB RAM.

We implemented test cases for all three applications (Section 3.4). PBRT uses tile rendering, so we do not create light subpaths for the whole image but per tile and iterate instead with the number of light subpaths equal to the number of camera samples per tile in each iteration. We used the default PRB setting (16×16), an optimization of this parameter is left for future work. Both camera subpaths and light subpaths are generated with only local importance sampling, i.e., BSDF sampling, without any guiding information, such as the position of their counterparts. This way we ensure that both sets are generated independently from each other and identically distributed within the same set.

We use a voxelization of $16 \times 16 \times 16$, resulting in a visibility map of 4096×4096 for all our test cases. For all tests, we initially run a regular



(a) Trace camera and light subpaths (b) Visibility map query

(c) Sample distribution (d) BiDir connection with MIS

Figure 3.3: An illustration of NEE++ in the context of BDPT. By constructing a distribution for all candidate light subpaths, which is a true subset of all light subpaths, we are able to utilize the precomputed visibility information and select a light subpath that has a higher chance of establishing successful connections with a higher probability.

render iteration and, in between two iterations, we create the visibility map. Timings for the construction are given in Table 3.1, which shows the negligible costs.

3.5.1 *Experiment Objectives*

Since we apply our technique in three different scenarios, three different objectives are considered. They are:

- Reducing shadowray tests for Section 3.5.2 while achieving similar results;

Algorithm 4 Pseudo code of NEE++ for Bi-Directional Subpath Sampling in a BDPT

```
1: procedure RenderIteration(scene, image, visMap)
2:     Ȳ ←LightTracing(scene, N)
3:     for i in range(M) do
4:         X̄_i ←RandomwalkFromSensor(scene)
5:         P_v^BiDir ←BuildDistribution(visMap, X̄_i, Ȳ)
6:         [Ȳ_j, p_v^BiDir(Ȳ_j)] ←Sample1D(Ȳ, P_v^BiDir, ξ)
7:         L_i ← Connect(X̄_i,Ȳ_j) / p_v^BiDir(Ȳ_j)
8:         image.add(L_i)
9: procedure BuildDistribution(visMap, X̄, Ȳ)
10:    subpathDistro← 0
11:    for all Ȳ_j in 𝕐 do
12:        subpathDistro[j] += queryPaths(visMap, X̄, Ȳ_j)
13:    return Normalize(subpathDistro)
14: procedure QueryPaths(visMap, X̄, Ȳ)
15:    v ← 0
16:    for all connections (x, y) in [X̄, Ȳ] do
17:        v += visMap.query(x,y)
18:    return v
```

- Robustly reducing variance comparing to other light sampling strategies (*Uniform, Power* and *Spatial*) for Section 3.5.3, given sample rate or runtime;
- Robustly reducing variance comparing to other unbiased estimators (*Path, (M)MLT* and *BDPT*) for Section 3.5.4, given sample rate or runtime.

3.5.2 *NEE++ Rejection Sampling Shadow Test*

We test our first technique (Section 3.4.1) in a unidirectional path tracer with the *Veach ajar* scene (Figure 3.4). Most of the light comes from the small gap at the door and the vast majority of the scene receives no direct light, which leads to many unnecessary shadow rays. The main objective of this experiment is to verify whether our technique could reduce shadowray tests (and subsequently respective runtime) while achieving similar results. Notice that plots in Figure 3.4 are roughly *equal quality*, of which the rMSE can be found in Table 3.2

This scene was mostly chosen for its particular configuration and as it is well-known. It has a particularly simple geometry, which results in the main cost being related to shading computations. In consequence,

Scene	*Test Corridor*	*Ajar*	*Kitchen*
Timing	231.4 ms	331.5 ms	401.2 ms
Scene	*Staircase*	*N. Classroom*	*Bathroom*
Timing	3413.4 ms	2350.1 ms	1579.6 ms

Table 3.1: Timing for caching visibility maps during our experiments. As a reference, the typical render time for these scenes with a relatively noise free output ranges from thousands of seconds to tens of hours.

one cannot expected a strong impact on the total rendering cost (our solution results in a $1-4\%$ speedup). Nevertheless, as can be seen in the figures and the table, we manage to cut around 80% of the shadow rays and use a fraction ($28-34\%$) of the runtime for visibility tests. The cutting of shadowrays is desirable for complex scenes.

3.5.3 *NEE++ Direct Light Sampling Strategy*

We test NEE++ as a light sampling strategy (Section 3.4.2) with the *Kitchen* scene (Figure 3.5) and the *Staircase* scene (Figure 3.7). Both scenes feature multiple light sources with varying intensity and size. Some sources are very small and located in a relatively complex region (e.g., above the microwave, under the staircase), which can lead to undesirable samples.

We compare our strategy against three existing light sampling strategies: *uniform distribution*, *power distribution* and *spatial distribution*. Uniform distribution treats each light source equally: $P(\text{Light}_i) = P(\text{Light}_j) = \frac{1}{\text{number of lights}}$. Power distribution favours light sources with larger area and/or stronger emitting intensity: $P(\cdot) \propto A(\cdot)E(\cdot)$.

These distributions are globally static distributions (invariant to the used query points). Spatial distribution takes distances as well as emitter surface and energy into account. Details about all distributions can be found in [85]. Results are given in Figure 3.5, Figure 3.6 and Figure 3.7. The timing and rMSE are reported in Table 3.3 and Table 3.4 respectively. We notice:

- NEE++ outperforms other strategies with the same sample rate in terms of error (Figure 3.5 and Figure 3.7);
- NEE++ better handles visibility-crucial regions and is less likely to sample strong light sources that are less visible (Figure 3.5, middle);

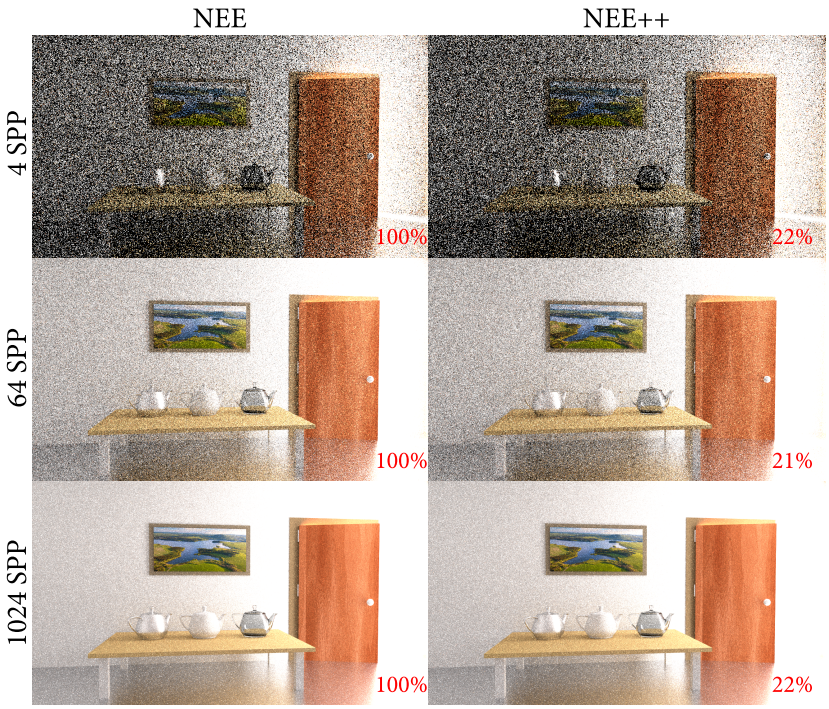


Figure 3.4: NEE++ Rejection Sampling Shadow Test. We achieve similar visual results with less shadow-ray queries and lower render time. Ratios of tests compared to regular path tracing are in the bottom-right corner of each image. Table 3.2 contains detailed statistics.

	rMSE		
	4 SPP	64 SPP	1024 SPP
NEE	5.82e−2	3.24e−2	2.12e−2
NEE++	5.91e−2	3.33e−2	2.18e−2
Ratio	1.01	1.03	1.03
	Shadowrays		
	4 SPP	64 SPP	1024 SPP
NEE	5.45e+6	8.72e+7	1.40e+9
NEE++	1.19e+6	1.92e+7	3.07e+8
Ratio	0.22	0.21	0.22
	Runtime Visibility Tests		
	4 SPP	64 SPP	1024 SPP
NEE	1.26s	5.47s	82.4s
NEE++	0.43s	1.54s	24.9s
Ratio	0.34	0.28	0.30

Table 3.2: Statistics for rMSE, shadow-ray queries and runtime for Figure 3.4. We achieve similar results (rMSE) with significantly reduced shadow rays (around 80% less) and much shorter runtime.

Sampl. Strategy	16 SPP	64 SPP	256 SPP	1024 SPP
Uniform	23.0s	105.6s	442.6s	1710.4s
Power	28.0s	118.9s	474.6s	1846.4s
Spatial	28.9s	128.6s	502.8s	1827.1s
NEE++	29.3s	132.8s	508.2s	1977.1s
Uniform	10.6s	50.2s	204.2s	768.7s
Power	10.2s	46.9s	184.8s	712.5s
Spatial	35.8s	77.3s	237.8s	855.3s
NEE++	11.5s	56.9s	231.3s	968.8s

Table 3.3: NEE++ as light sampling strategy (Figure 3.5 and Figure 3.7). Statistics for runtime. Top: the *Kitchen* scene; bottom: the *staircase* scene. NEE++ has a slight 1% $\sim$ 8% runtime overhead compared to the other three strategies with the same number of samples.

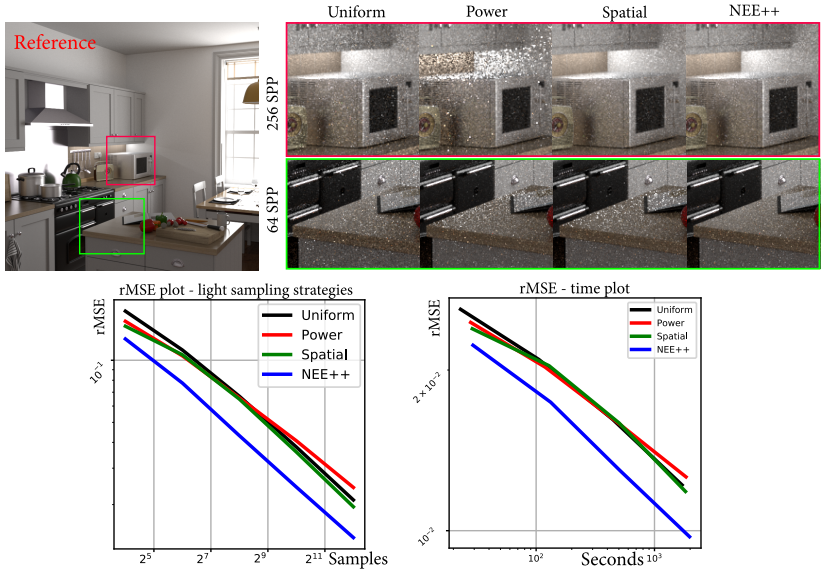


Figure 3.5: NEE++ Direct Light Sampling Strategy. Our strategy is more robust with $\sim$ 20% variance reduction overall. Details are given in Table 3.3 and Table 3.4. Figure 3.6 shows an equal variance comparison.

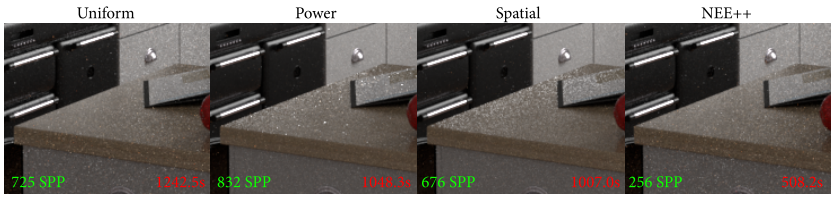


Figure 3.6: Equal rMSE comparison corresponding to Figure 3.5. Sample rate is given in green, while the runtime performance is red. For similar variance, our method takes 35%, 30% and 38% the sample rate of *uniform*, *power* and *spatial* respectively with runtime ratios of 40.1%, 48.5%, and 50.5%.

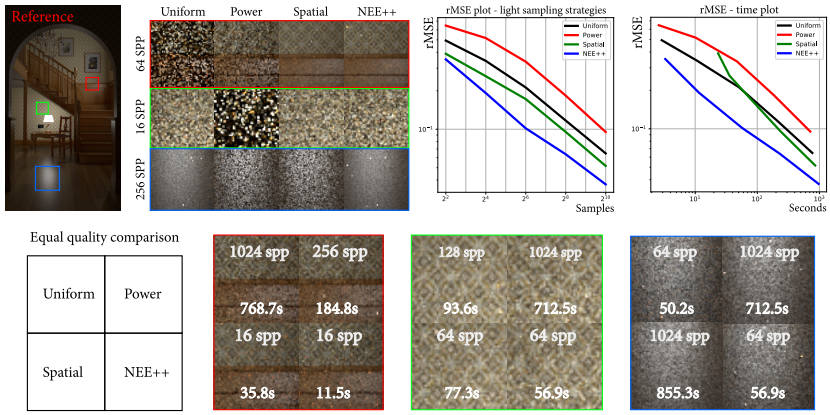


Figure 3.7: Top left: *Staircase* reference. Top centre: equal sample rate comparison. Top right: plot of rMSE against sample rate and runtime. Bottom: equal quality comparison. NEE++ outperforms other methods in terms of convergence and is more robust.

Strategy	16 SPP	64 SPP	256 SPP	1024 SPP
Uniform	2.60e−2	2.09e−2	1.62e−2	1.22e−2
Power	2.46e−2	2.03e−2	1.60e−2	1.26e−2
Spatial	2.40e−2	2.04e−2	1.59e−2	1.18e−2
NEE++	2.23e−2	1.74e−2	1.29e−2	9.74e−3
Uniform	3.43e−1	2.12e−1	1.17e−1	6.47e−2
Power	5.17e−1	3.38e−1	1.83e−1	9.52e−2
Spatial	2.60e−1	1.72e−1	9.52e−2	5.16e−2
NEE++	1.92e−1	1.02e−1	6.37e−2	3.70e−2

Table 3.4: NEE++ as light sampling strategy (Figure 3.5 and Figure 3.7). Statistics for rMSE. Top: the *Kitchen* scene; bottom: the *Staircase* scene. Overall, NEE++ has a $\sim$ 20% improvement with the same number of samples.

- NEE++ achieves 20% variance reduction with 1% $\sim$ 8% computational overhead (Table 3.3);
- NEE++ requires only about a fraction of the number of samples compared to other light sampling techniques to match the error (Figure 3.6 and Figure 3.7).

The power distribution fails to handle the regions above the microwave, and causes severe fireflies due to the low probability of reaching this light source. The spatial distribution creates fireflies on the table for the same reason. The runtime overhead could be even further reduced by caching and reusing light distributions, which we do not do in our current implementation. We notice an unexpectedly long runtime for strategy *Uniform*, our repeated experiments confirm this behaviour. We use the original code from PBRT [85] and the exact mechanism behind this phenomenon is not known to us.

3.5.4 *NEE++ Bi-Directional Subpath Sampling*

We compare NEE++ using BDPT Subpath Sampling (Section 3.4.3) with three unbiased MC estimators: unidirectional path tracing (Path), bi-directional path tracer (BDPT) and *Metropolis light transport* (MLT). We test all methods on three scenes: *Test Corridor*, *Nasty Classroom* and *Bathroom*. All scenes feature complex visibility: *Test Corridor* exhibits a minimum connected path depth of 4 (Figure 3.8), *Nasty Classroom* has many light sources with blocking geometry (Figure 3.10). Statistics for rMSE are provided in Table 3.5.

TEST CORRIDOR This is a typical scenario where traditional MC methods, such as ours, have a strong disadvantage compared to Markov chain MC (i.e., MLT). All traditional MC methods struggle to connect the camera to the light source, given the nature of the scene configuration. In fact, this scene should not exhibit a single black pixel upon convergence but all connected light rays have to undergo several bounces. The render results and the runtime are given in Figure 3.8, with stats plotted in Figure 3.9. For the same sample rate, we observe:

- NEE++ outperforms all other methods in terms of zero-radiance pixel ratio (Figure 3.9 (right));
- MLT has better rMSE, at the expense of uneven convergence (Figure 3.9 (left));
- At the same sample rate NEE++ has a $\sim$ 15% runtime overhead compared to BDPT (Figure 3.9) while achieving 20% error reduction;

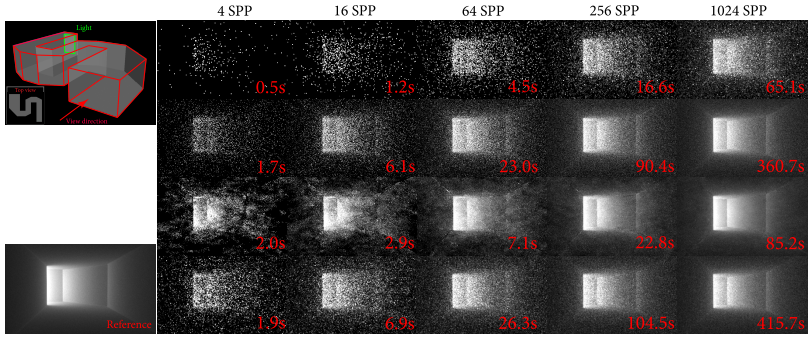


Figure 3.8: Left top: *Test Corridor* scene. The whole corridor geometry is only open at the right end and closed elsewhere. Left bottom: reference, rendered using unidirectional path tracing with 102,400 samples per pixel. Notice that none of the pixels is black. Right: Results for various methods. Notice the difficult visibility configuration, which poses a challenge for all methods. Path tracing struggles to reach the light source, causing many occluded shadow-ray queries. BDPT manages to connect the light source to the sensor but a large portion of the pixels remain black. MLT successfully exploits the vicinity in path space, once a connected path is constructed, resulting in much less zero-radiance pixels. The uneven convergence is an inherent problem. Our NEE++ manages to establish more connected paths than BDPT, while achieving more even convergence than MLT with the same number of samples. Render times are indicated in the bottom-right corner.

- Path tracing performs worst in terms of rMSE and zero-radiance pixel ratio per sample (Figure 3.9).

The fact that NEE++ has less zero-radiance pixels illustrates the benefit of our solution and the effectiveness of finding even complex paths. Further, the runtime overhead with respect to BDPT is low, while achieving a stronger error reduction in this scene, despite it being a very disadvantageous scene for our approach. While MLT does outperform all competitors in terms of error, MLT is less often employed in practice due to its irregular convergence, which becomes more visible in the following scenes.

NASTY CLASSROOM AND BATHROOM We use a modified version of the *Classroom* scene, using a night scenario with light emitting geometry on the ceiling instead of the original virtual sun and sky light. Challenging parts of the scene include classroom furniture with glossy materials, the cabinet of desks that are hidden away from direct light sources, and the area below the desks, where visibility towards light sources is complex. Render results, rMSE plot, and runtime analysis are given in Figure 3.10, Figure 3.12 and Table 3.5. Equal quality comparisons

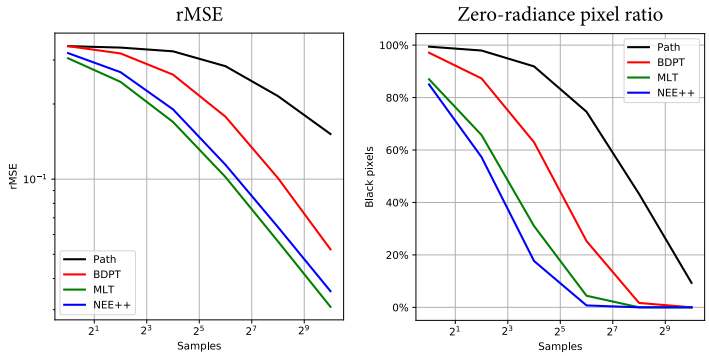


Figure 3.9: Statistics of the *Test Corridor* scene (Figure 3.8): rMSE and zero-radiance pixel ratio. MLT performs slightly better in terms of rMSE than NEE++ but it suffers from uneven convergence and more black pixels. To get more out of Figure 3.8 and the plots here, we see that at a roughly equal error rate (10^{-1}): Path tracing takes 2048 samples and 123 seconds; BDPT 256 samples and 90 seconds; NEE++ 84 samples and 36 seconds: a 67% sample-rate reduction and 60% in render time.

are given in Figure 3.11. When comparing at the same sample rate, we observe:

- NEE++ outperforms all other methods given the same render time;
- NEE++ improves over BDPT by 15% $\sim$ 20% variance reduction with $\sim$ 7% runtime overhead
- MLT shows very irregular convergence behaviour, largely due to the complex visibility and glossy materials

When comparing equal quality, we observe that NEE++ typically takes a fraction of the sample rate ($50 - 1.5\%$) and runtime ($49.9 - 14\%$).

3.6 DISCUSSIONS

Throughout all our experiments, we see the effectiveness of NEE++. We tested multiple application scenarios and showed improvements over existing methods: in Section 3.5.2, we see the reduction in visibility tests; in Section 3.5.3, we see how applying our solution to the visibility term in the rendering equation, we are able to perform better direct illumination sampling; in Section 3.5.4, we see a similar but extended version of direct sampling, for which we are able to connect camera subpaths to better candidate light subpaths given the visibility information.

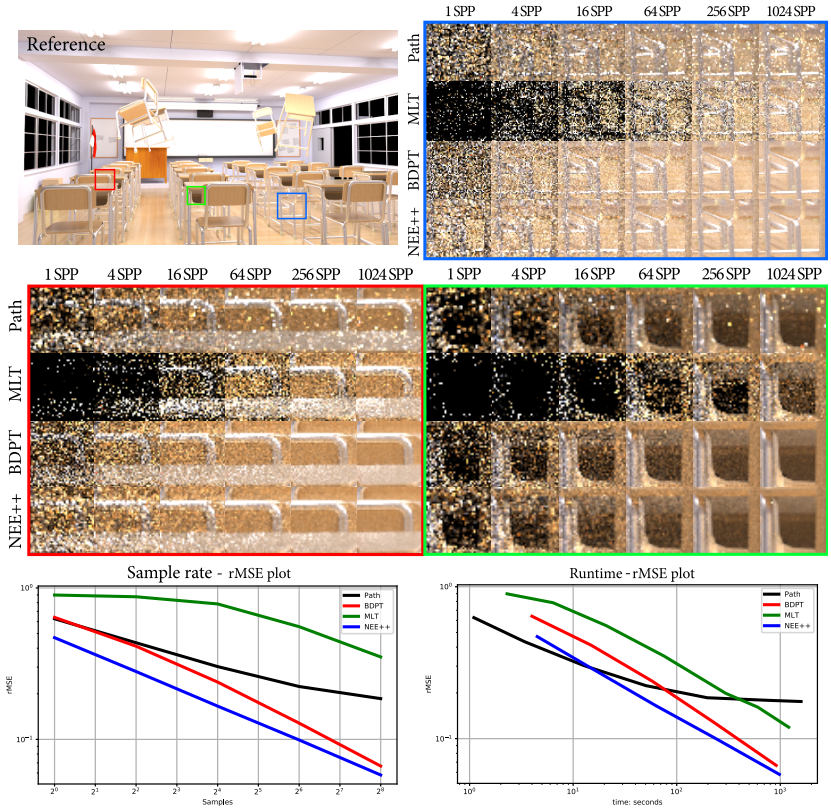


Figure 3.10: Top left: *Nasty Classroom* reference. Bottom left: rMSE plot. Bottom right: experiment timings.

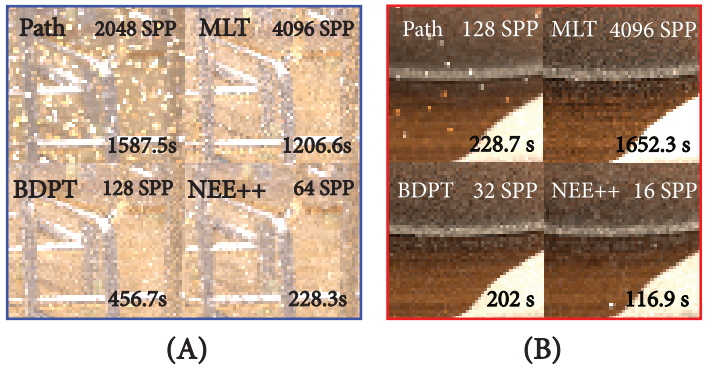


Figure 3.11: Equal quality comparison (A) for scene *Nasty Classroom* in Figure 3.10 and (B) for scene *Bathroom* in Figure 3.12.

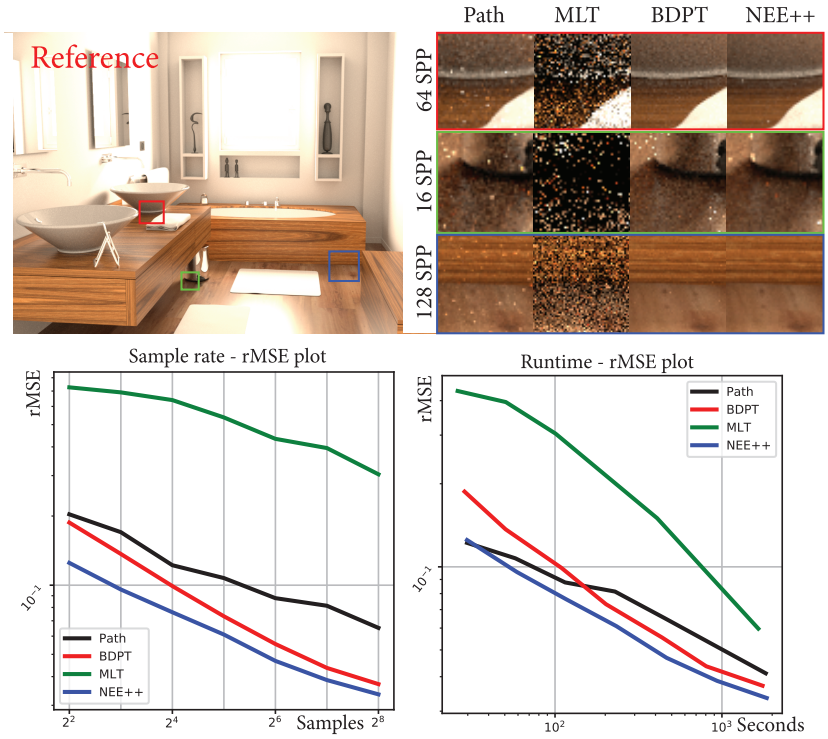


Figure 3.12: Results for the *Bathroom* scene. rMSE is plotted against sample rate and runtime.

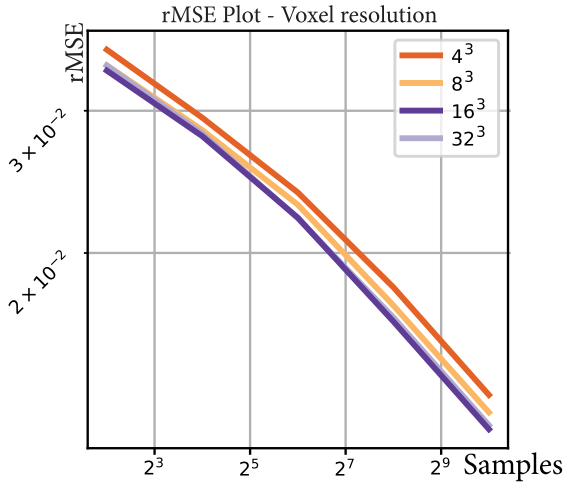


Figure 3.13: Comparison of different voxel-grid resolutions for the *Kitchen* scene (Figure 3.5). The overall error is reduced with increasing visibility-map resolution but a sweetpoint exists, depending on the scene complexity (compare 16^3 and 32^3). Legend: grid resolution.

Method	4 SPP	16 SPP	64 SPP	256 SPP
Path	3.36e−01	3.25e−01	2.83e−01	2.15e−01
MLT	2.45e−01	1.69e−01	1.02e−01	5.63e−02
BDPT	3.19e−01	2.62e−01	1.78e−01	1.01e−01
NEE++	2.68e−01	1.90e−01	1.14e−01	6.42e−02
Path	4.35e−01	3.03e−01	2.23e−01	1.86e−01
MLT	8.75e−01	7.84e−01	5.55e−01	3.50e−01
BDPT	4.12e−01	2.39e−01	1.28e−01	6.67e−02
NEE++	2.81e−01	1.66e−01	9.91e−02	5.82e−02
Path	2.03e−01	1.22e−01	8.78e−02	6.52e−02
MLT	7.26e−01	6.39e−01	4.34e−01	3.04e−01
BDPT	1.87e−01	9.91e−02	5.55e−02	3.71e−02
NEE++	1.25e−01	7.62e−02	4.68e−02	3.35e−02

Table 3.5: NEE++ for Bi-Directional subpath sampling (Figure 3.8, Figure 3.10 and Figure 3.12). Statistics for rMSE. Top: *Test Corridor*; Middle: *Nasty Classroom*; Bottom: *Bathroom*. Overall, our method manages to improve $\sim 20\%$ upon BDPT at the same sample rate. The poor robustness of MLT could also be clearly derived by comparing the different performance in three test scenes.

The improvements are achieved at the cost of a small computational overhead. Per sample costs occur when accessing the visibility map queries and its building process, which could be further optimized, adds an additional cost. Nevertheless, with respect to the overall gain or total computation times, these costs are negligible. Further, we could envision reducing these costs by caching or reusing information, which we do not do in our current implementation - all examples generated the visibility map from scratch.

In most cases, we manage to achieve better convergence per sample, mostly by constant improvement over existing MC methods, i.e., roughly at the same convergence rate but better by a constant offset. Even with the computational overhead and without further optimization, we can conclude that our technique outperforms the other techniques when it comes to fix-runtime performance given the nature of the MC convergence rate being $1/\sqrt{N}$.

3.6.1 *Grid resolution*

While we are working on a relatively low grid resolution, one might argue that a visibility map with higher resolution gives higher precision and therefore better results. However, in our experiments, we find that increasing the resolution increases memory requirements and preprocessing time but quickly shows little benefit (Figure 3.13 shows an example in the *Kitchen* scene).

Higher resolution poses other challenges such as high memory footage and overfitting. Our primary experiments for different grid resolution at high sample rate show no conclusive evidence about the improvement. Our way of stochastically using the values might contribute to this phenomenon.

Given the cost-benefit at current low computational overhead, we leave further investigation for future work.

3.6.2 *Compatibility with Path Guiding*

As mentioned in Section 3.2, our visibility mapping could be used along with various path guiding techniques. Most path guiding methods rely on crafting distributions for importance sampling during path construction. Such distributions usually target one or more terms in the rendering equation. In the same spirit as in Section 3.4.2, we could further incorporate our visibility mapping in assisting processes where rays are guided by, e.g., photons, since visibility is usually not taken care of. Additionally, our visibility Russian roulette is always an option for pruning shadow ray tests.

3.6.3 *Comparison with PCBPT*

Our technique resembles PCBPT [82] in some aspects. As both methods target better connections during BDPT. Yet, our method works on subpath level while PCBPT works on path vertex level. This induces three main differences.

- First of all, most of the cached information for our technique comes for free in a previous and independent pass, while for PCBPT a dedicated procedure to test visibility between a subset of camera vertices and all light vertices in current pass is needed repeatedly. This difference not only affects runtime, but the domain. Our visibility cache is a spatial-spatial relation while PCBPT caches only spatial-index relations. This difference makes it possible for our technique to be used in various scenarios while PCBPT is only applicable to BDPT;

- Secondly, we select a whole light subpath based on the visibility map while for PCBPT at each camera vertex a completely different light subpath could be chosen. This further complicates the calculation for multiple importance sampling. In this sense PCBPT is a derivation of *Combinatorial BDPT*[70] while our technique is an extension for standard BDPT;

- Thirdly, PCBPT runs iteratively and as a result the computational overhead is much higher than ours ($\sim$ 20% as reported in [82]).

3.6.4 *Limitations*

During our experiments, we noticed that while our method mostly improves the result given a fixed sample budget or runtime, our method will be less efficient in scenes where visibility is not the major source of complexity (e.g., scenes with large open spaces). Furthermore, we currently englobe the entire scene in a voxel grid, which does not cover directional light or environment maps.

3.6.5 *Future work*

Using a uniform voxelization wastes some memory with empty voxels and the coarse resolution does not always well represent the underlying geometry. This could be addressed with adaptive voxelization but it comes at the expense of further pre-processing and query time. Additionally, a geometry-based adaptation alone will not be successful, as geometry density is not necessarily the cause of complex visibility.

Therefore it is not clear if this would improve the technique. Currently, our solution does already avoid calculations on empty voxels and we have seen that a higher resolution does not yield much additional benefit in our test scenes. Also, a ray traversal in the acceleration tree structure would have to be replaced with a query in a tree structure, although one might be able to make use of hashing schemes.

One potential solution for adaptive structure could be to make use of hierarchical division where submaps are associated to spatial divisions of the scene. I.e., creating several level of visibility map and only querrying at the correct level in the hierarchy. One other potential solution is to make use of an existing acceleration structure: e.g., *bounding volume hierarchy* (BVH). This volume information comes for free but requires further indexing before it can be compatible with our pipeline.

In this chapter, we investigate the possibilities of unbiased usage of NEE++, but unbiasedness might not always be of highest priority. For example, in the context of *many light rendering*, sharing one visibility map across *virtual point light* (VPL) sources instead of rendering one shadow map for each VPL could be useful. In realtime applications, visibility tests could become the bottleneck as such shading operations are typically cheap. Our technique could serve as an approximation in situations where inaccurate visibility won't pose fatal error. It could also be an option for secondary and onward bounces in path tracing to use visibility-map values directly, since for interactive purposes primary bounces make the most visual impact.

3.7 CONCLUSIONS

In this chapter, we presented NEE++, a visibility mapping technique with applications to several rendering problems. It efficiently caches and reuses visibility information to improve light transport simulations. By collecting visibility in a matrix-like form during a low-cost preprocess, we can reduce visibility tests by up to 80% and prefer tests that are likely to succeed instead. Variance at the same time is also reduced by ~20%. Our solution is unbiased and general. It is easy to implement and integrate into existing pipelines. The exemplary cases show that it can be widely applied in several scenarios but many more potential applications exist.

Part II

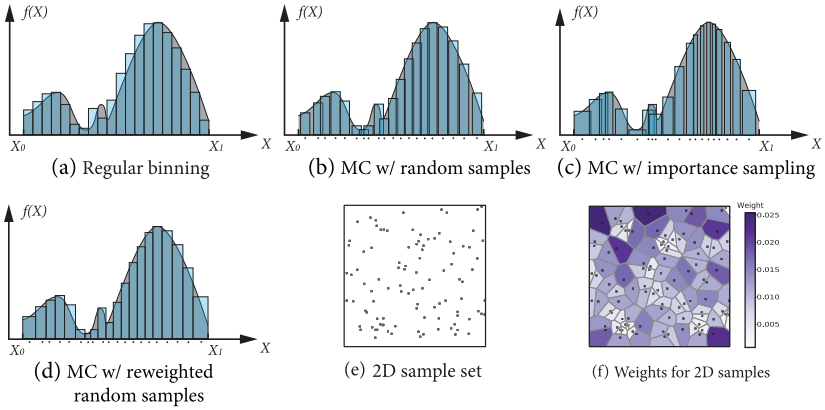
RECONSTRUCTION

4

GEOMETRIC SAMPLE REWEIGHTING FOR MONTE CARLO INTEGRATION

Abstract

Numerical integration is fundamental in multiple Monte Carlo rendering problems. We present a sample reweighting scheme, including underlying theory, and analysis of numerical performance for the integration of an unknown one-dimensional function. Our method is simple to implement and builds upon the insight to link the weights to a function reconstruction process during integration. We provide proof that our solution is unbiased in one-dimensional cases and consistent in multi-dimensional cases. We illustrate its effectiveness in several use cases.



(a) Regular binning (b) MC w/ random samples (c) MC w/ importance sampling

(d) MC w/ reweighted random samples (e) 2D sample set (f) Weights for 2D samples

Figure 4.1: Top row: Three integration methods using the same amount of function evaluations (i.e., 20 samples): (a) Riemann sum through regular binning (according to right-side value) (b) MC integration using uniform random samples; (c) MC integration using samples distributed according to a pdf w.r.t. function value. Notice that in (a) and (b), the associated bin widths are equal, i.e., $\frac{1}{20}$. Bin widths in (c) are adjusted according to the density linked to the sample generation. Notice the overlaps/gaps between sample bins as illustrated in (b) and (c). Illustrations of our methods: (d) uniform random samples with our reweighting. Notice the absence of gaps/overlaps and bin widths being adjusted according to sample positions. (e) a set of 10^2 i.i.d. random samples in $2D$, $(0,1)^2$. Rather than directly averaging all samples (i.e., $weight = 10^{-2}$ for all samples), our consistent estimator uses the Voronoi volume as sample weight during the reconstruction process (f). Notice that the weights here are inversely proportional to the density.

4.1 INTRODUCTION

Monte Carlo (MC) techniques form the foundation of realistic image synthesis for decades [5]. The principle is simple: a function is sampled and the samples are combined to approximate its integral. Standard MC is often referred to as *brute-force* because of the easy implementation and guaranteed convergence to the correct solution, but often suffers from high variance and slow convergence.

The integral approximation can be improved by reconstructing the underlying function from the samples and previous work focused on several particular cases (e.g., shadows [60] or depth of field [65]). In this chapter, we focus on one-dimensional numeric integration and revisit the function reconstruction process.

Our observation is that standard sample weights are often less accurate for lower sampling rates because they do not reflect the integration domain nor the local sample density well. Our contribution is to pro-

pose a weighting scheme that considers all samples of a given set and defines weights based on their Voronoi cells. This results in a consistent estimator that outperforms standard weighting schemes. Additionally, we render the estimator unbiased for sets of independent and identically distributed (i.i.d) uniform random samples. While the proof of unbiasedness is complex, the algorithm is easy to implement.

While our work focuses on the one-dimensional case, the derived theory shows that the bias of the consistent estimator is expected to be low. We can therefore expect also good performance for low-dimensional problems and provide a numerical evaluation.

Our solution can be easily integrated in existing rendering pipeline. It provides an immediate benefit over existing schemes for various rendering problems and builds upon a solid theoretical analysis.

After covering prior work and MC integration (Section 4.2), we motivate our approach (Section 4.3) and explain the core of our method (Section 4.4). We then evaluate its numerical performance and show applications to rendering (Section 4.5) before concluding (Section 4.7).

4.2 BACKGROUND

MC METHODS Since the 80s [5], MC integration plays a major role in rendering complex distribution effects, such as motion blur, depth of field, soft shadows and global illumination. The complete light transport is described by the *rendering equation* [7], which can be solved using *path tracing* as an associated MC solution.

Standard Monte Carlo (MC) methods solve a definite integration $I = \int_\Omega f(x)\mathrm{d}x$ of a function f over a finite support $\Omega \subset \mathbb{R}^d$ by using a random sample set ($\{x_i \in \Omega\}$). The resulting estimator is

$$\hat{I}_{MC} = \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{4.1}$$

Nevertheless, not all samples taken during the evaluation of an integral contribute strongly to the result.

IMPORTANCE SAMPLING The sampling process can be influenced with a probability distribution function (pdf) $p : \Omega \to \mathbb{R}$ [19]. Its unbiased estimator is:

$$\hat{I}_p = \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \tag{4.2}$$

which effectively weighs samples differently, given non-uniform sample input.

Importance sampling is interesting when having knowledge about the scene. For instance, importance sampling the light source works

better in scenes with small or point light sources [48, 54]. Sampling according to the BSDF works better with glossy to highly glossy surfaces[12, 25]. Multiple importance sampling (MIS) combines different sampling strategies [19].

One strategy to modify subsequent sample choices is to rely on previous samples, i.e., a Markov process. Metropolis sampling [29] can handle complex light-path configurations by extensively exploring contributing paths once they are discovered. Multidimensional k-d trees [55] can update a global structure with samples, which can then be used as a means to control future sample placement.

REWEIGHTING Specialized weight definitions have proven beneficial for rendering, e.g., derived in Sobolev spaces [96]. However, these previous solutions target hemispherical illumination integrals and are not generally applicable to other problems. A reweighting scheme was also proposed for addressing firefly artifacts [100] but the solution is biased and limited to narrow application scenarios.

Recently Bitterli *et. al.* present a reweighting process following resampling from reservoir sampling (ReSTIR) [103]. Reservoir sampling [6] generates fix size samples that follows certain population distribution. The reweighting as proposed in the work is biased but leads to significant variance reduction.

Other specialized reconstruction techniques exist, including solutions for soft shadows [60], defocus blur [65], and motion blur [68], which lead to significant improvements. More complex reconstructions for environment maps illumination [38] and light fields [69] have proven very successful but are biased (though consistent). Our method handles general functions and is independent of the application scenario.

General weighted reconstruction during MC integration has initially been investigated: a trapzoidal rule and a nearest-neighbor rule have been proposed by Yakowitz *et. al.* [2]. Our novelty lies in supporting domains with open boundaries and a new weight definition. This is different from using the Euclidean distance between nearest neighbors [2]. Additionally, our method results in an unbiased estimator. As this approach pursues similar goals, we will analyze the difference in Section 4.4.1.

Our approach links sample weights to *Voronoi* cell sizes. The latter has also been studied in the context of anti-aliasing problems [10], for which the two-dimensional area of Voronoi cells bounded by the pixel are directly used as sample weights. This leads to improved anti-aliasing, but the theory has not been further developed for unbiased solutions, nor generalized to other contexts. Voronoi cell size has been used as weights for Monte Carlo integration by Vorechovsk *et. al.* [88]. They treat domain boundaries by either clipping to them or by extending them periodically by adding auxiliary samples next to the opposing

domain border. While both solutions improve numerical performance of MC integration, we show that such a solution is biased in Section 4.4.

4.3 FORMULATION AND PROBLEM STATEMENT

Riemann integration approximates an integral using function values $f(x_i)$ times a weight $w(x_i)$:

$$\hat{I} = \sum_{i=1}^{N} w(x_i) f(x_i). \tag{4.3}$$

The Riemann weights are constant and stem from a partitioning of the support Ω into equally-sized *hypervolumes* - in 1D, these are intervals - each containing exactly one sample at the center. Referring to Equation 4.2, the estimator using samples following distribution p is also a sum of function values $f(x_i)$ times a weight $w(x_i) = \frac{1}{Np(x_i)}$, which are typically easy to compute but cannot be associated to a hypervolume partitioning of Ω. They are scaled versions of $\frac{1}{N}$ according to a per-sample density $p(x_i)$; corresponding volumes would overlap or introduce gaps. Only when the sample count is increased to densely cover the support Ω, due to the stochastic nature of the process, the difference in the weight definitions becomes negligible. See Figure 4.1 (a), (b) and (c) for an illustration. In consequence, especially for low sample counts, the weights $\frac{1}{Np(x_i)}$ do not lead to a good function approximation[46].

4.4 GEOMETRIC SAMPLE REWEIGHTING

Our goal is to associate integration weights to samples that relate to an improved function reconstruction. We will first define a consistent solution, inspired by Riemann integration (Section 4.4.1). It is independent of the sampling pattern and can be applied on any sample set in a postprocess to improve the integral approximation. While consistent, it is not unbiased for all sampling strategies (Section 4.4.2). We then propose a modification to obtain an unbiased estimator for 1D cases (Section 4.4.3). We treat uniform random samples that are *independent and identically distributed* (i.i.d.). Section 4.4.4 discusses the possibility to generalize our method for samples generated with a non-uniform pdf, and we propose a simple solution for piecewise-constant importance sampling, which is not uncommon in the context of computer graphics.

4.4.1 *Consistent Estimator*

Riemann integration typically assumes a regular partitioning of the domain. Using a *Voronoi tessellation* of the sample points, it is possible to

partition the domain Ω to take the actual sample density into account. A Voronoi tessellation is a partition into regions such that the points in each region share the same closest sample location. It can be shown that the Voronoi cell corresponds to the intersection of half spaces defined by hyperplanes that are equidistant to two sample points. The theory of Voronoi diagrams is beyond the scope of this chapter but details can be found in works by Aurenhammer *et. al.* and De Berg *et. al.* [11, 24].

For a D dimensional problem setting, the diagram will be bounded by the hypercube $(0,1)^D$, the domain from which samples are drawn. The volume of each Voronoi cell determines the corresponding sample weight. Given that the cells are intersections of half-spaces, these Voronoi cells are convex and their volume can be easily computed. The resulting estimator of this approach is:

$$\hat{I}_C = \sum_{i=1}^{N} w_C(x_i) f(x_i), \text{where } w_C(x_i) = \frac{|V_i|}{|\Omega|} \tag{4.4}$$

Such an estimator is consistent, as this construction implicitly approximates the integrand via a piecewise-constant representation. Figure 4.1 (d) and (f) show an illustration of this strategy for a one-dimensional and two-dimensional case, respectively. In principle, even more advanced approximations could be used. However, as we will show, it turns out that such weight definitions, while consistent, lead to a biased estimate. In the following, we will show the reasons and derive an unbiased estimator for i.i.d. uniform sampling and stratified sampling for our use case with $D = 1$.

4.4.2 *Bias Analysis*

The problem of integrating over a *given* set of samples brings some changes to the classic MC integration process: given a set of samples $\{x_i | x_i \sim \mathcal{U}(0,1)\}$, we want to work with a specific set $\{x_i\}$ such that $0 < x_1 < x_2 < \cdots < x_N < 1$. This will ease the derivation of the weights in our method. From *Order Statistics* [42], we know that the probability of a set fulfilling this condition is $P(\{x_i\}) = N! \prod_{i=1}^{N} p(x_i)$. Intuitively, a random set of samples with N elements can be ordered in $N!$ ways.

To ease understanding of our bias analysis, we first illustrate the derivation with classic MC $\hat{I}_{MC}$ (i.e., equal weight for all samples). We denote the composed operation as $\mathbb{F}(\{x_i\}) = \sum_{i=1}^{N} \frac{1}{N} \frac{f(x_i)}{p(x_i)}$. Conveniently, $P(\{x_i\}) = N!$ and $\mathbb{F}(\{x_i\}) = \sum_{i=1}^{N} \frac{1}{N} f(x_i)$ for uniform sam-

pling ($p(x) = 1$). The expectation of integration using such a set of samples is:

$$\begin{aligned}
\mathbb{E}\left[\hat{I}_{MC}\right] &= \int_{(0,1)^N} \mathbb{F}(\{x_i\}) P(\{x_i\}) \mathrm{d}\{x_i\} \\
&= \int_0^1 \int_0^1 \cdots \int_0^1 \left[\sum_{i=1}^{N} \frac{1}{N} \frac{f(x_i)}{p(x_i)}\right] N! \prod_{i=1}^{N} p(x_i) \mathrm{d}\{x_i\} \\
&= \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \left[\sum_{i=1}^{N} \frac{1}{N} f(x_i)\right] N! \mathrm{d}\{x_i\} \\
&= N! \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \\
&\left[\frac{1}{N} f(x_1) + \frac{1}{N} f(x_2) + \cdots + \frac{1}{N} f(x_N)\right] \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1 \\
&= \int_0^1 f(x) dx,
\end{aligned} \tag{4.5}$$

Line 2 expands from line 1 following the definition. Properties $p(x_i) = 1$ and $0 < x_1 < x_2 < \cdots < x_N < 1$ are used to get line 3 from line 2. The integration domain is denoted as $(0,1)^N$ to indicate domain of the sample vector. The notation $\mathrm{d}\{x_i\}$ means the differential sample set, which expands to $\mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1$. In Section a.1, we prove the last step, showing that classic MC is indeed unbiased.

A similar derivation for our consistent estimator (Equation 4.4) yields:

$$\begin{aligned}
&\mathbb{E}\left[\hat{I}_C\right] = \\
&N! \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \\
&\left[\frac{x_1 + x_2}{2} f(x_1) + \sum_{i=2}^{N-1} \frac{x_{i+1} - x_{i-1}}{2} f(x_i) + \right. \\
&\left. \left(1 - \frac{x_{N-1} + x_N}{2}\right) f(x_N)\right] \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1
\end{aligned}$$

$$
\begin{aligned}
&= N! \cdot \int_0^1 \Bigg\{ \frac{1}{2} \cdot \frac{x \cdot (1-x)^{N-1}}{(N-1)!} \\
&\quad + \frac{1}{2} \cdot \frac{(1-x)^{N-1}\left[(N-1)\cdot x + 1\right]}{N!} \\
&\quad + \sum_{i=2}^{N-1} \left[\frac{1}{2} \cdot \frac{(1-x)^{N-i}\left[(N-i)\cdot x + 1\right] \cdot x^{i-1}}{(N-i+1)!(i-1)!} \right. \\
&\quad \left. - \frac{1}{2} \cdot \frac{(1-x)^{N-i} \cdot x^i}{i(N-i)!(i-2)!} \right] \\
&\quad + \left[\frac{x^{N-1}}{(N-1)!} - \frac{1}{2} \cdot \frac{x^N}{N(N-2)!} - \frac{1}{2} \cdot \frac{x^N}{(N-1)!} \right] \Bigg\} \cdot f(x)\mathrm{d}x \\
&= \int_0^1 \frac{(N - Nx - x)x^{N-1} + (Nx + x - 1)(1-x)^{N-1} + 2}{2} \cdot f(x)\mathrm{d}x.
\end{aligned}
\tag{4.6}
$$

Unfortunately, the expression is not identical to $\int f(x)\mathrm{d}x$ for $N > 1$. Thus, our consistent estimator as introduced in Equation 4.4 introduces bias. Similarly, we can show that other correction schemes using augmented periodic samples, as proposed in [88], results in a biased estimate (compare Section a.2). The issue is that a weight adaptation of the boundary samples is insufficient, as inner samples also share a weight expectation different from $1/N$. Nevertheless, as the bias is linked to the expression $(N - Nx - x)x^{N-1} + (Nx + x - 1)(1-x)^{N-1}$, we do observe that for the large part of the domain it converges to 1 for larger N, which illustrates that the consistent estimator can be of practical relevance.

4.4.3 *Deriving an Unbiased Estimator*

A closer look reveals that Equation 4.6 takes the form of $\int g(x) \cdot f(x)$, where $g(x) = [(N - Nx - x)x^{N-1} + (Nx + x - 1)(1-x)^{N-1} + 2]/2$. As $g(x) \neq 0$ for all $x \in (0,1)$, we can define $F(X) = \frac{f(x)}{g(x)}$. Following the derivation of Equation 4.6 but considering F instead of f, we obtain the expectation of this new estimator $\hat{I}_C^F$:

$$
\begin{aligned}
\mathbb{E}\left[\hat{I}_C^F\right] &= \int_0^1 g(x) \cdot F(x)\mathrm{d}x = \int_0^1 g(x) \cdot \frac{f(x)}{g(x)}\mathrm{d}x \\
&= \int_0^1 f(x)\mathrm{d}x = \mathbb{E}\left[\hat{I}\right].
\end{aligned}
\tag{4.7}
$$

It can be seen that sampling F yields an estimate for the integration of f. Hence, the function g can serve as a weight definition to obtain an

unbiased estimator for the integrand $f(x)$ over a set of N i.i.d. uniform samples with reweighting. Consequently, our unbiased estimator is given as:

$$\hat{I}_{GR} = \sum_{i=1}^{N} w_{GR}(x_i) f(x_i) = \sum_{i=1}^{N} \frac{|V_i|}{|\Omega|} \frac{1}{g(x_i)} f(x_i) \tag{4.8}$$

4.4.4 *Working with Non-uniform Samples*

The general form of MC integration follows from Equation 4.5, where $p(x_i)$ is not necessarily 1 for all x_i. Thus, the analytic expression for when deriving it as in Equation 4.6 leads to uniform samples depending on p. As p can be arbitrary, it is impossible to develop the expression further.

Following order statistics, we need to know the *joint probability density* of samples following given distribution. For uniform distributions this is convenient as $p(x_i) = 1$. For non-uniform cases, deriving corresponding g_p term would mean the presence of $\prod_{i=1}^{N} p(x_i)$ that can not be easily cancelled out. For this to be unbiased, strictly analytical approach is required. Even for a given p with simple analytical formulation, chances are high that an analytic integration is usually too complex. Nevertheless, it is possible to adapt the uniform approach to the special case of p being a piecewise-constant distribution, which is not uncommon in many practical use cases. Here, we can stratify the domain according to the distribution and reweight samples within each stratum, following the previous solution for the whole domain. In this way, we can estimate the final integration without introducing bias. We provide a numerical evaluation of this strategy in Section 4.5.1.

4.5 RESULTS

In Section 4.4, we developed the unbiased estimator for one-dimensional functions (Equation 4.8). Nevertheless, we also illustrated that the use of Voronoi volumes remains consistent for arbitrary dimensions. As the computation of a Voronoi diagram can be costly for higher dimensions, the method is best suited for low-dimensional functions. In the following, we will use the unbiased estimator for one-dimensional and the consistent estimator for multi-dimensional problems. The derivation of an unbiased reweighting scheme for arbitrary dimensions is an open challenge for future work.

The result section is organized in two parts, first covering the numerical performance (Section 4.5.1) of our estimator before applying it to several rendering problems (Section 4.5.2). All tests compare multiple different estimators:

1. Standard MC (i.i.d. uniform sampling)
2. Our weighted standard MC (i.i.d. uniform sampling)
3. Stratified MC (stratified sampling)
4. Our weighted stratified MC (stratified sampling)
5. Low discrepency sequences when applicable

4.5.1 *Numerical Performance*

We first test the numerical performance of MC integration in 1D and 2D. The 1D functions were chosen to include discontinuities, large-scale variations, and small scale changes. In the supplementary material, the reader can find many more test cases for our algorithm. Here, due to their representative behavior, we analyze:

$$f(x) = 10 \times \begin{cases} \sqrt{-x^2 + 0.5x} & x \leq 0.25 \\ -\sqrt{-x^2 + x - 0.1875} + 0.25 & 0.25 < x \leq 0.5 \\ 20 \times (x - 0.5) & 0.5 < x \leq 0.55 \\ 1.0 & 0.55 < x \leq 0.65 \\ -20 \times (x - 0.7) & 0.65 < x \leq 0.7 \\ 0.1 \times \sin(10\pi \cdot (x - 0.7)) & 0.7 < x \leq 0.8 \\ 0.25 \times \sin(10\pi \cdot (x - 0.8)) & 0.8 < x \leq 0.9 \\ 0.5 \times \sin(10\pi \cdot (x - 0.9)) & 0.9 < x \end{cases}$$

For the 2D consistent estimator, we take the *Cameraman* image (see Figure 4.3 (d)). The results are shown in Figure 4.3(3). Generally, the MSE drops as more samples are added (Column 1). Our solutions outperform standard uniform sampling (and stratified sampling) by several orders of magnitude and converges around 10^3 (10^2) times in 1D and 10^2 (10) times faster in 2D.

We study the numerical performance of non-uniform samples that are importance sampling the piecewise-constant distribution in Figure 4.2 (actual function $f(x)$ is plotted in dashed lines). We take two reasonable non-uniform piecewise constant distributions. Red: 4-piecewise distribution that does not match $f(x)$ and blue: 10-piecewise distribution that matches $f(x)$ perfectly. It can be seen from the results that both reweighted estimators show similar/superior converging behaviors while a distribution that does not match function yields no practical benefits comparing to uniform samples. Unexpectedly, the distribution that matches the function perfectly only slightly outperforms uniform and a non-matching distribution. Meanwhile, all three

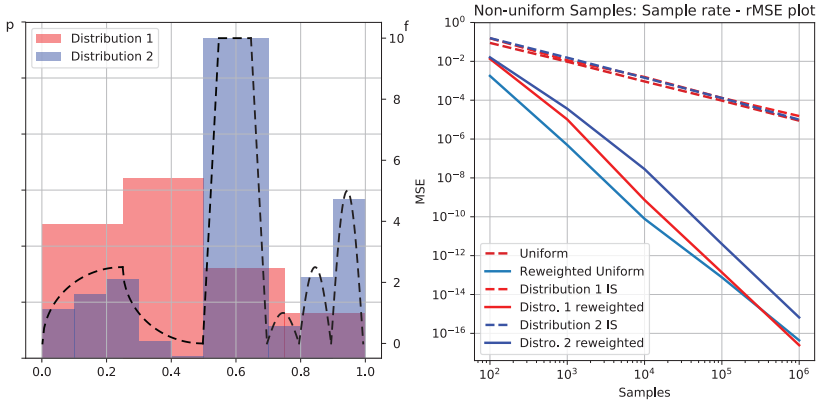


Figure 4.2: Numerical performance of reweighting both uniform and two non-uniform distributions. Integrand in test is plotted in dashed line in the left subfigure. Uniform samples are drawn normally while non-uniform samples are drawn from the piecewise constant distributions to the left. Log-log scale convergence performances are plotted in the right subfigure.

reweighted version show similar convergence behavior. This illustrates not only the effectiveness of our reweighting but also how tricky importance sampling is in real-world scenarios dealing with functions that have significant/sharp variations. It could also be seen that in this specific experiment, a carefully crafted distribution is much less effective when it comes to variance reduction when comparing to our reweighting scheme.

Numerical performance against common 1D low discrepency sequences is presented in Figure 4.3 (a). As can be seen from the result, our reweighted uniform/stratified achieves better performance than traditional stratified and *Latin Cube* sequence while slightly worse than *Halton* and *Sobol* sequence. For 2D cases (Figure 4.3 (c)), both reweighted estimators perform the best and all low discrepency sequences fail to make any improvement.

We next investigate the impact of distributing samples into batches to show the unbiased convergence behavior of our reweighting scheme in 1D cases. First, we fixate the amount of samples to 10^6 (Figure 4.3(b)). As expected, the performance of standard uniform sampling remains invariant with respect to the amount of samples per batch, as it is already an averaging process. In general, more batches means a higher overall sample count and all methods improve with the addition of batches. In all cases, the graphs stop after reaching 10^6 samples. Our solution performs best and the graphs illustrate a fast convergence.

In a direct comparison, our estimator asymptotically achieves the same convergence rate (with a constant offset) as the estimator pro-

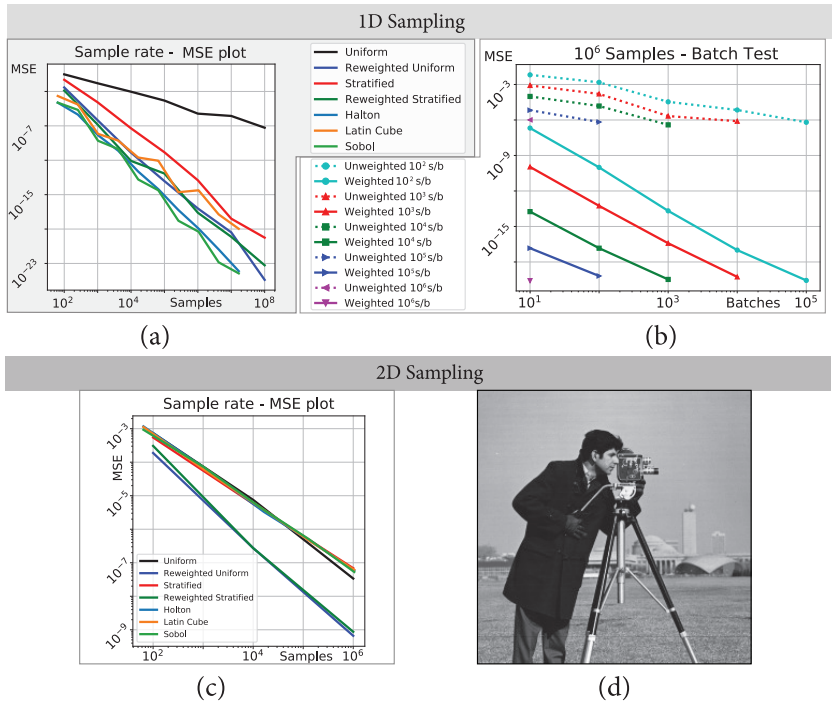


Figure 4.3: We apply our geometric sample reweighting to one and two dimensional MC integration problems. Subfigures (a) & (b) show results for 1D case, and (c) shows a 2D integration problem where the underlying function is an image shown in (d). Performance gain of our reweighting scheme can be observed from the faster convergence rate comparing to unweighted ones in 1D cases and all methods in 2D cases. In 1D cases, our unbiased reweighting achieves performance that is close to low discrepency sequence (better than latin hypercube). In 2D cases, our consistent estimator outperforms all methods in test, including all low discrepency sequences. All results are plotted in log-log scale.

posed in [2], which is proven to be $O(N^{-4/D})$, where D is the dimension of the function. The interested reader is referred to the original work for the proof. Yet, in contrast, our work has the advantage of being unbiased. We used the same functions and configurations as in their original work. The results are plotted in Figure 4.4.

4.5.2 *Application to Rendering*

We implemented our method in Mitsuba [66], targeting one and two-dimensional integration problems, namely motion blur (Section 4.5.2.1), dispersion (Section 4.5.2.2), depth of field (Section 4.5.2.3) and illumination integrals (Section 4.5.2.4). We evaluate MSE and visual appearance, as well as convergence behavior. For all implementations, our reweighting operates at a per-pixel level. Notice that some low discrepency sequences are incompatible with certain integrators in Mitsuba. For example, only (0,2) sequence can be used in motion blur tests and Hammersley can not be used for direct illumination integrator.

4.5.2.1 *Motion Blur*

To simulate motion blur, distribution rendering samples the time domain: For a pixel (i, j), the luminance $L_{(i,j)}$ is given by:

$$L_{(i,j)} = \int_{t_{\text{open}}}^{t_{\text{close}}} f_{(i,j)}(t)\mathrm{d}t,$$

with t_{open} and t_{close} being the shutter opening and closing time and f incorporating the shutter function. The problem is thus one-dimensional and we tested our implementation in two animated scenes (Figure 4.5 and Figure 4.6).

All temporal samples are drawn uniformly from $[t_{\text{close}}, t_{\text{open}}]$. In both scenes, objects are moving at high speed. The implementation takes minimum effort as only one outer loop in the program needs modification.

4.5.2.2 *Spectral Rendering*

Light dispersion can happen at reflective or refractive dielectric materials, leading to rainbows, resulting from different wavelengths traveling in different directions. It requires spectral sampling to simulate multiple wavelengths. To reduce the complexity of the additional spectral dimension[58], *hero wavelength spectral sampling*[79] can be used as an approximation:

$$\hat{I}_{(i,j)} = \frac{1}{N}\sum_{k=1}^{N}\frac{f_{(i,j)}(\lambda_k)}{p_{(i,j)}(\lambda_k)}$$

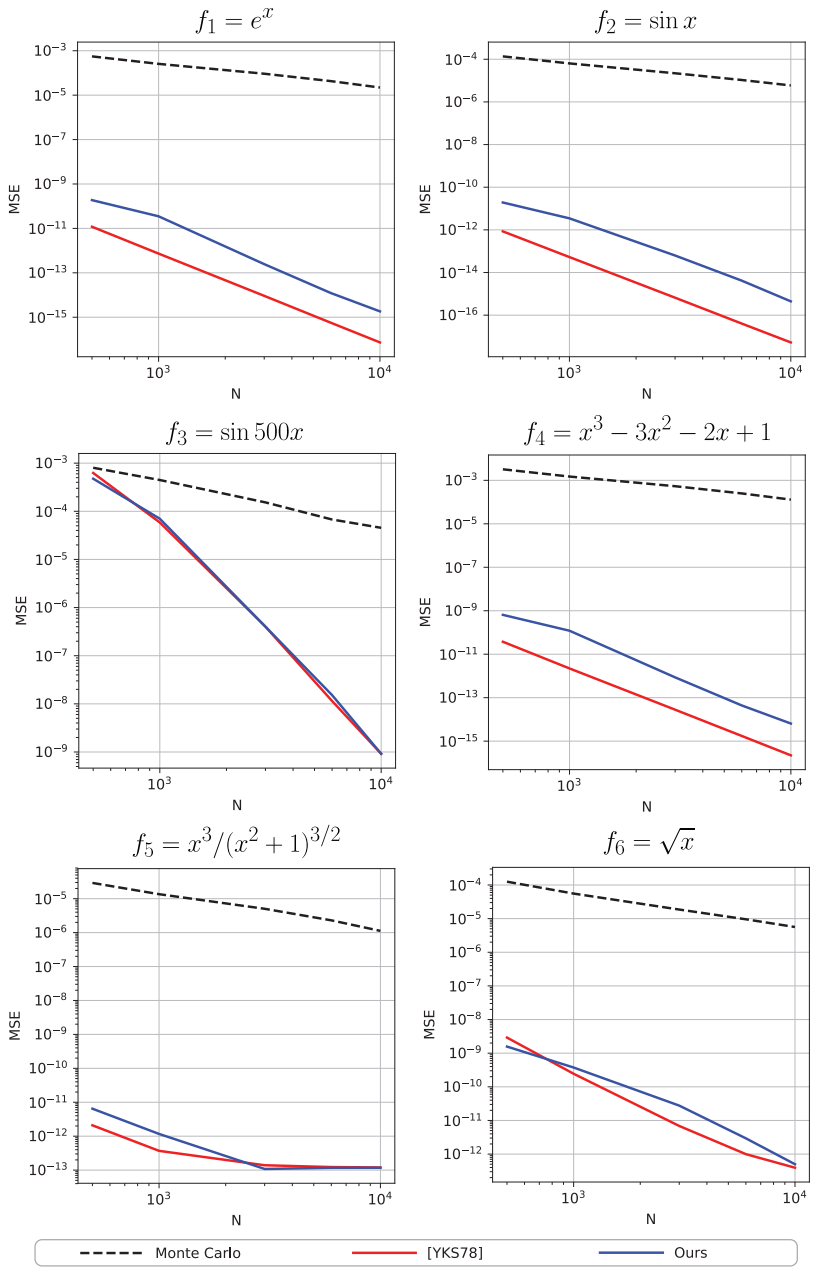


Figure 4.4: Log scale MSE plot of three estimators: Monte Carlo, [2] and ours. As can be seen from the slope, our estimator is asymtotically parallel to [2] and is much superior to Monte Carlo. In some cases, our estimator performs less as good, but that is extra cost of remaining unbiased.

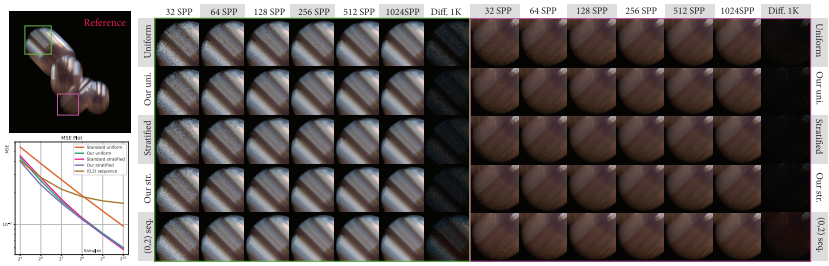


Figure 4.5: Four highly glossy spheres moving in different directions. In each subfigure: corresponding renderings of increasing sample rate, difference of 1K sample rate with reference. A much brighter difference (with reference) can be observed for (0,2) sequences, which confirms the MSE plot at the bottom left. All results are plotted in log-log scale.

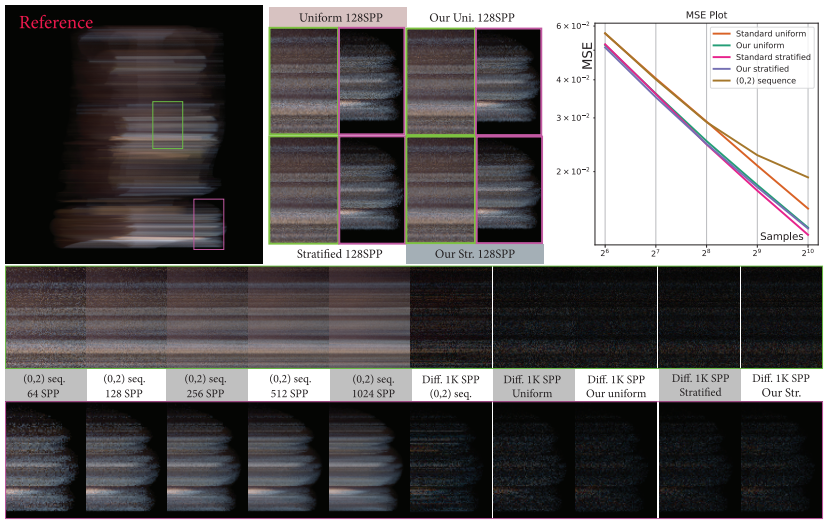


Figure 4.6: Highly glossy Buddha moving horizontally with 128 samples per pixel. Bottom two rows: renderings of corresponding regions for (0,2) sequences and the 1K sample rate difference of all methods. Similar behavior is observed for (0,2) sequences as in Figure 4.5. All results are plotted in log-log scale.

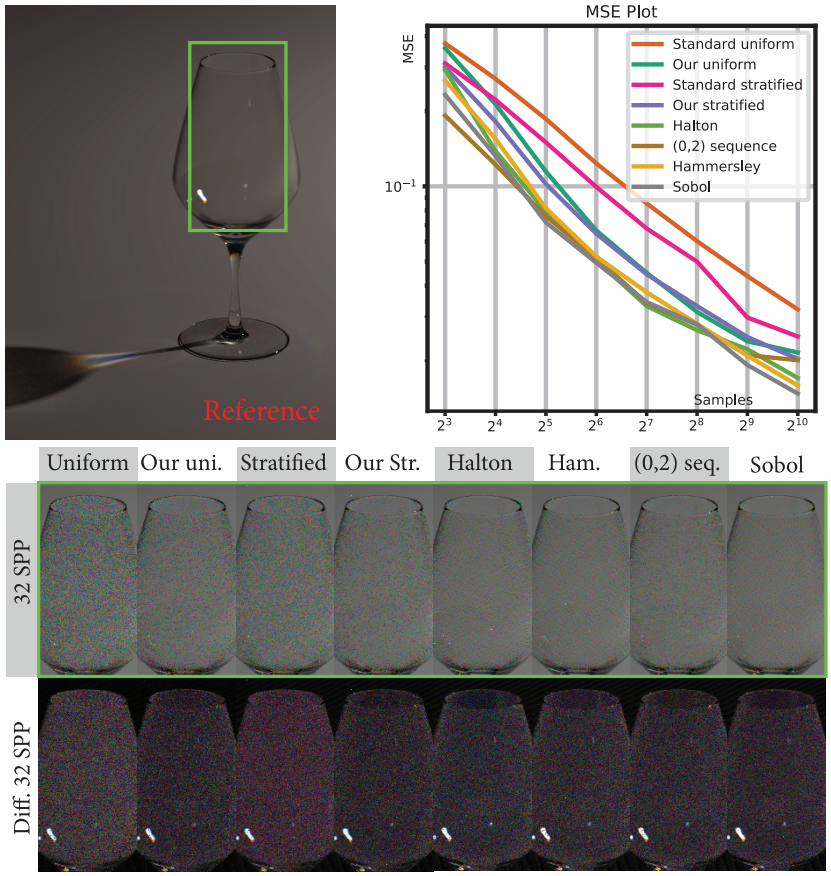


Figure 4.7: Wineglass with dispersive dielectric materials with 32 samples per pixel. As can be seen, our reweighted schemes improve upon uniform and stratified samples and produces much smoother result in terms of color noise. Significant improvements can be observed in low discrepency sequences. All results are plotted in log-log scale.

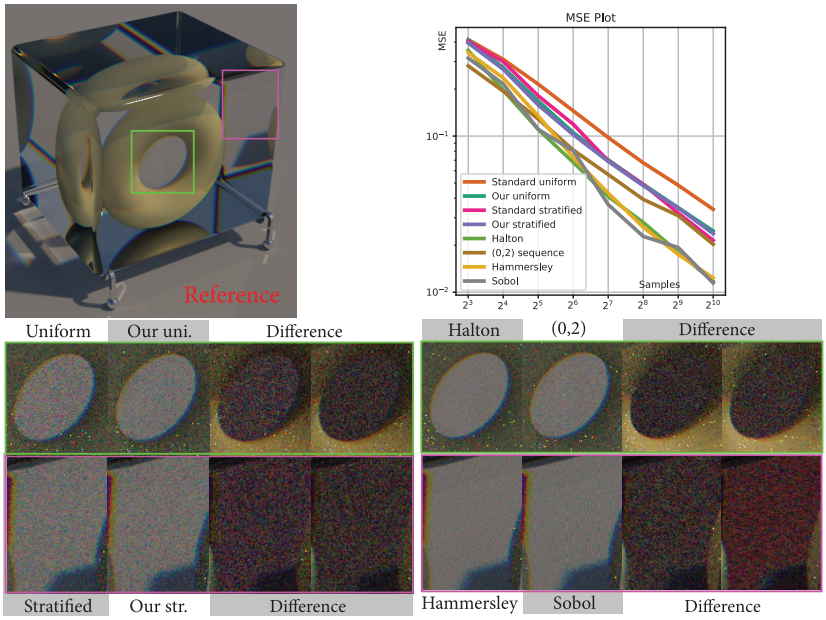


Figure 4.8: Torus with dielectric materials with 64 samples per pixel. Our reweighted schemes produce smoother renderings while avoiding aliasing that is obvious in low discreoency renderings. All results are plotted in log-log scale.

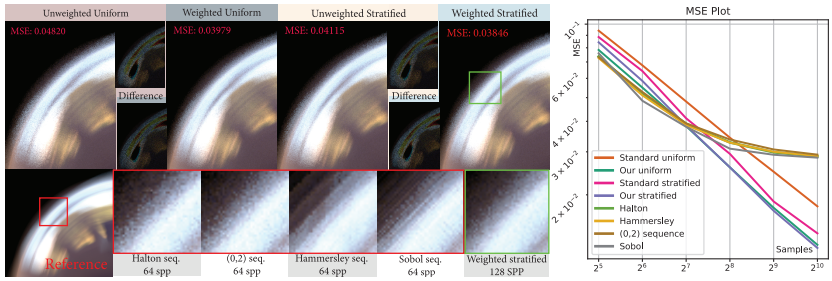


Figure 4.9: Glossy sphere crossing the focal plane of the camera. Top: 128 samples per pixel; bottom: equal error renderings of low discrepency sequences. Lower error can be observed in low discrepency sequence renderings, at the cost of noticeable correlation patterns. All results are plotted in log-log scale.

Our implementation of spectral sampling uses 15-bin wavelengths. Hero wavelength sampling is used with 3 shifted additional wavelength samples [79]. We tested our method with two scenes configured with dispersive dielectric materials (Figure 4.7 and Figure 4.8). As shown in the results, our method brings down color noise significantly and dispersive regions look much smoother at a low sampling rate.

4.5.2.3 *Defocus Blur*

A camera with aperture leads to defocus blur/depth-of-field effects. The aperture is usually modelled as a 2D shape, e.g., a square, a circle, or a star. The shaper is sampled to determine the origin of each primary sample ray. These rays pass through the positions on the focal plane corresponding to the pixels. For lens aperture $\mathbb{A} \subset \mathbb{R}^2$, we obtain:

$$L_{(i,j)} = \int_{\mathbb{A}} f_{(i,j)}(s)\mathrm{d}s.$$

To determine our weights, we use a 2D Voronoi diagram based on the aperture samples. We tested a simple glossy sphere illuminated by an environment map (Figure 4.9).

4.5.2.4 *Direct Illumination*

Leaving out irrelevant terms, the luminance $L_{\mathbf{x}}$ at a scattering point $\mathbf{x}$ with one bounce is given by:

$$\begin{aligned} L_{\mathbf{x}} &= L_e(\mathbf{x}) + L_{\text{direct}} + L_{\text{indirect}} \\ &= L_e(\mathbf{x}) + \int_{\mathbb{L}} f_s(\mathbf{x}) L_e(l \to \mathrm{x})\mathrm{d}l + \int_{\Omega} f_s(\mathbf{x}, \omega) L_i(\omega)\mathrm{d}\omega, \end{aligned}$$

where L_e denotes light emission and $l \in \mathbb{L}$ denotes all light sources. In this application, we use light sampling instead of random rays to ensure

that the light source is always sampled. Our reweighting achieves the best convergence and, as shown in the insets, also the smoothest results (Figure 4.10∼Figure 4.12). Each image uses 4 primary rays per pixel and at each scattering event 256 light samples. Our solution is applied to these 256 samples.

4.6 DISCUSSION

In all cases, our solution leads to smoother visual result and less black pixels in the falloff regions. It improves uniform sampling and stratified sampling comparing to unweighted ones. From the MSE plots, we can see that standard MC with uniform sampling has the worst performance, while our weighted stratified sampling generally performs best. In some cases where the integrand is smooth, stratified sampling alone achieves good results. We can also see that even with uniform sampling as input, our weighted uniform sampling not only improves upon the unweighted version but also has a performance comparable to our weighted stratified sampling.

When comparing tests with low discrepency sequences (Halton, (0,2), Hammersley and Sobol sequences), we observe the general trend of seemingly biased convergence, as is evidently shown in Figure 4.5, Figure 4.6, Figure 4.9 and Figure 4.10∼Figure 4.12, where the MSE plot curve shows flattening trend towards higher sample rate. This is partly due to the fact that the correlation patterns (errors) usually take huge amount of samples to converge, as such low discrepency sequences (*quasi Monte Carlo, qmc*) work by trading off bias and variance [30]. Moreover, most renderings with Halton, Hammersley and Sobol sequences show strong correlation/aliasing in images. In low sample rate cases, lower error could be achieved with low discrepency sequences, at the cost of obvious correlation/aliasing.

Due to the fact that we pre-compute samples and weights, the runtime for unweighted and weighted rendering are identical and therefore not reported. Pre-computing sample weights has a negligible overhead (e.g., on a 8-core CPU: less than 2 seconds for 1024 one dimensional samples in Figure 4.7, which takes few hours to render at a resolution of 512×512; around 5 milliseconds overhead per pixel for 1024 two dimensional samples in Figure 4.10∼Figure 4.12. One can always precompute weights and reuse for different integrals, e.g., pixels in different render tiles.

4.7 CONCLUSION

Our reweighting scheme enables a better approximation than standard MC weights. It is general and does not require any prior knowledge

about the Riemann integrable function. Implicitly, it approximates the integrand via a reconstruction from the samples. We presented an analysis of the introduced bias of a consistent estimator based on Voronoi cell volumes and propose an unbiased 1D estimator. We showed the practical benefit of our solution for various rendering problems.

Despite the system error introduced for high-dimensional integration, we observe huge improvement in terms of variance reduction. Nevertheless, it requires the computation of high-dimensional Voronoi cell volumes, which can be costly. Still, we demonstrate several low-dimensional application cases in this chapter.

As our method makes no assumptions about the function to be integrated, it is an attractive approach for many purposes and can be easily integrated in existing solutions (e.g., ReSTIR [103]). We believe our method could be of interest beyond graphics community.

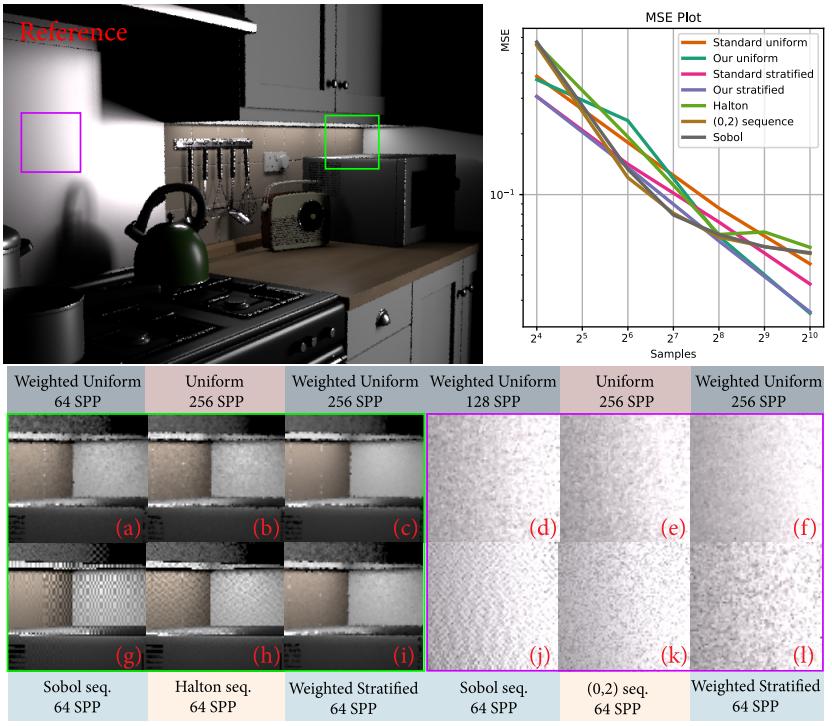


Figure 4.10: An equal quality and sample rate comparison of applying our reweighting in the *Kitchen* scene. In subfigures (b)&(e): 256 samples per pixel with unweighted uniform; in (a)&(d): equal error rendering (comparing to (b)&(e)) is achieved with weighted uniform at 64&128 samples per pixel; in (c)&(f) weighted uniform results in a much smoother rendering at the equal sample rate of 256 samples per pixel (comparing to (b)&(e)); subfigures (g)-(l): equal error renderings of low discrepency sequences and the weighted stratified rendering. As can be observed from the results, our weighted schemes delivers results close in quality with lower sample rates and better converged results at equal sample rate. Despite the low error renderings, low discrepency sequences result in obvious patterns of correlation. All results are plotted in log-log scale.

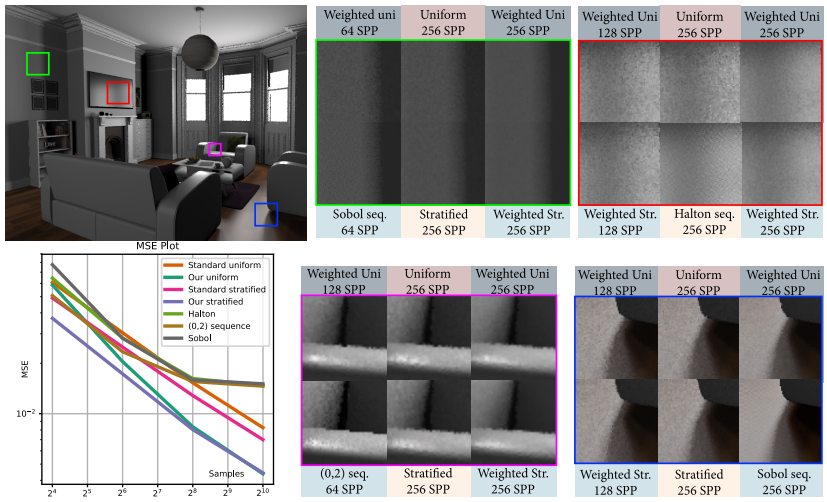


Figure 4.11: An equal quality and sample rate comparison of applying our reweighting in the *Livingroom* scene. As can be observed from the results, our weighted renderings achieve better convergence at equal sample rate comparing to unweighted ones and use less samples reaching similar error level. Noticeable patterns of correlation are observed in low discrepency sequence renderings. All results are plotted in log-log scale.

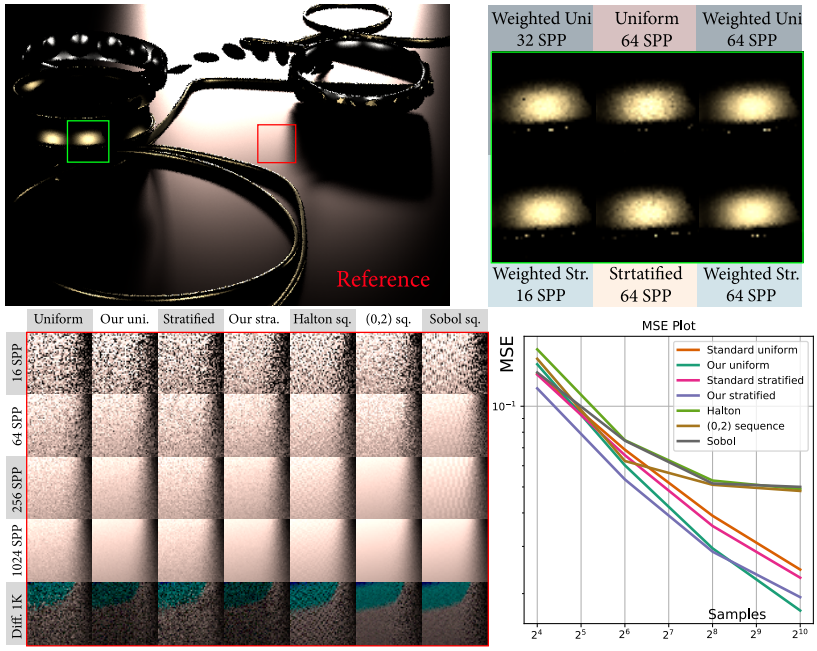


Figure 4.12: An equal quality and sample rate comparison of applying our reweighting in the *Necklace* scene. Similar behavior can be observed as in Figure 4.11. (0,2) sequences manage to produce alias-free renderings. However, brighter difference comparing to reference is observed, which confirms the MSE plot in the right subfigure. All results are plotted in log-log scale.

Part III

MODELLING

5

EFFECTIVE COC: EXTENDED APERTURE MODEL FOR PERSPECTIVE CAMERAS IN RENDERING

Abstract

We present an extended camera lens model for screen-space depth-of-field simulations. Building upon a geometric analysis, it produces plausible out-of-focus effects, including distortion and shrinking. The approach is easy to implement and integrates well as a postprocess in rendering pipelines inducing only a small overhead. While our focus is on real-time rendering, we show that the insights can be applied to accelerate high-quality offline solutions as well.

5.1 INTRODUCTION

Depth-of-field (DoF) effects are an important element of realistic image synthesis in offline and real-time rendering. Ray-optics models produce sharp representations for in-focus objects and defocus blur for out-of-focus objects. The amount of defocus blur is described by the radius of the *circle of confusion* (CoC). The size of the CoC increases the further an object is away from the plane of focus.

The main problem addressed in this chapter is the phenomenon of *lensing* and *distortion* seen through blurred fallout overlaps when out-of-focus objects are partially occluded by other out-of-focus objects. These effects arise naturally in ray-tracing methods, which explicitly model the camera lens including aperture. Nevertheless, they have not yet been addressed in real-time rasterization-based DoF methods that approximate lens blur via image filters.

In this chapter, we revisit the commonly used DoF methods and analyze the lensing and distortion effects induced by partial occlusions. We extend the model for real-time rendering by introducing the concept of *effective CoC*. We implement the new model in screen space to recreate the distortions and lensing effects without significantly increasing the computational cost of screen-space DoF effects. Finally, we show that these principles can also be used to accelerate offline rendering by influencing the sampling process based on the derived effective CoC.

Specifically, our contributions are:

- An extended DoF model, considering lensing and distortion;
- A real-time screen-space algorithm incorporating these effects;
- A high-quality offline-rendering adaptation.

5.2 BACKGROUND

Many modern rendering methods build on the model of an ideal pinhole camera. In such models, the rendered image is perfectly sharp, with no objects seemingly out of focus. In the real world, no such ideal systems exist. A more realistic approach involves modeling the cameras with a thin lens system [32], having a finite focal length and an aperture. For a given imaging plane, a corresponding plane of focus exists on the other side of the lens (Figure 5.1). Points on the plane of focus project perfectly onto the image plane. Points on either side of the plane of focus project to an area. The *circle of confusion* (CoC) corresponds to this $2D$ area that a $3D$ point in space produces on the $2D$ image plane.

CLASSIC THIN LENS MODEL The qualititive properties of the classic lens model above is described by the *Guassian lens equation*, which states

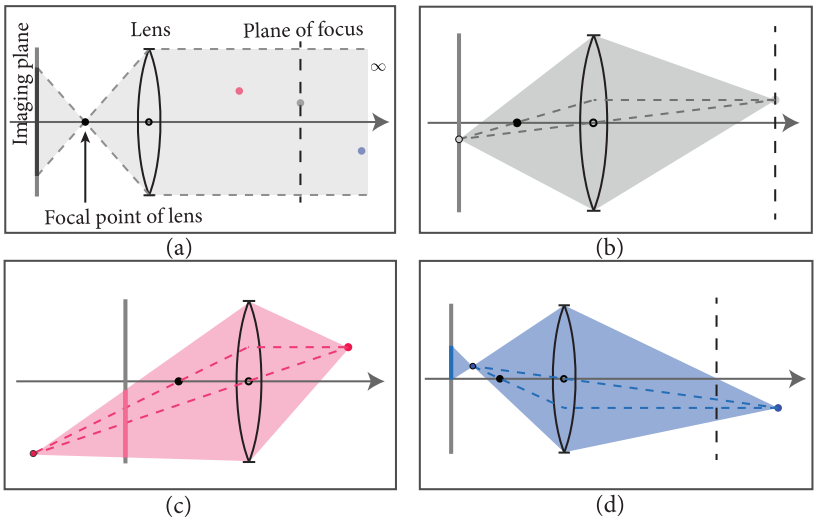


Figure 5.1: Classic thin lens model as commonly presented in optics literature. (a) An object at infinity can be considered to cast parallel rays through the lens, which are focused at the focal point of the lens. The distance of the imaging plane to the lens influences the size of the location on the film that the rays intersect. (b) The grey object is on the plane of focus. Its image is perfectly sharp. (c) The red object is closer to the camera than the plane of focus. The focus point is beyond the imaging plane, leaving a circle on the film. (d) The blue object is further away than the plane of focus. The image here is also out of focus, due to the fact that the focus point lies in front of the film. The focus point is traced with dotted lines according to the principles of lens optics. Only points on the plane of focus map to points on the image plane, otherwise, we always have a CoC of finite size.

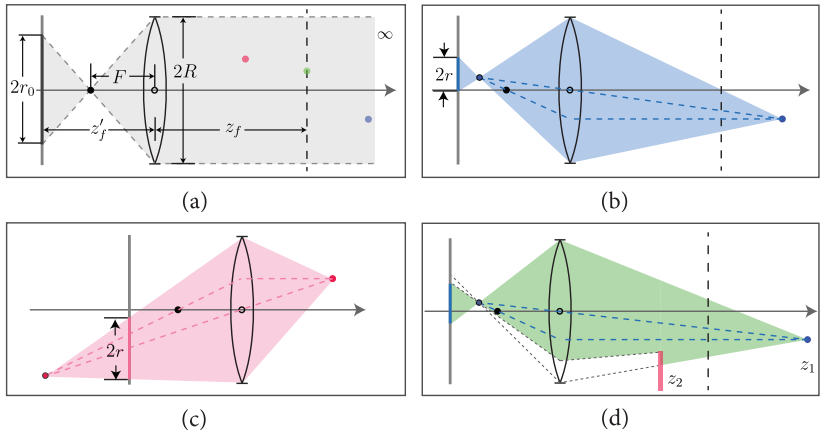


Figure 5.2: Corresponding size of circle of confusion with classic lens model. (a) Objects at infinity focus on the focal point and leave the CoC size of $2r_0$. (b) and (c) Out of focus objects leave a finite sized CoC on the imaging plane. (d) The phenomenon of interest in this chapter.

that for lens with focal point F, the image depth z' of object at depth z is given by the following equation:

$$\frac{1}{z'} - \frac{1}{z} = \frac{1}{F}. \tag{5.1}$$

With reference to Figure 5.2 (a), we denote the radius of the lens system as R, which is typically implemented with aperture featuring a variable size (i.e. different f-stops). The imaging plane is placed at z'_f, which corresponds to the depth of z_f where the plane of focus is. The relationshiop between z_f and z'_f is again guarded by Equation 5.1. Plug in the equation to the cases as illustrated in Figure 5.2 (b) & (c), we can get corresponding sizes of CoC:

$$r = \frac{|z' - z'_f|}{z'} R, \tag{5.2}$$

which goes to 0 as z goes to z_f from both directions (z_f^- and z_f^+), approaches $r_0 = \frac{|z'_f - F|}{F} R$ as z goes to ∞; and goes to ∞ as z goes to 0. Notice here we use term depth to represent the *absolute* distance between a point of interest and the lens centre, z is therefore not a signed term.

OCCULUSIONS The configuration of occulusions naturally implies a difference in depth, i.e., the occuluder (at depth z_2) is closer to the camera than the occuluded object (at depth z_1): $z_2 < z_1$. Such configuration could include the following five cases when taking the

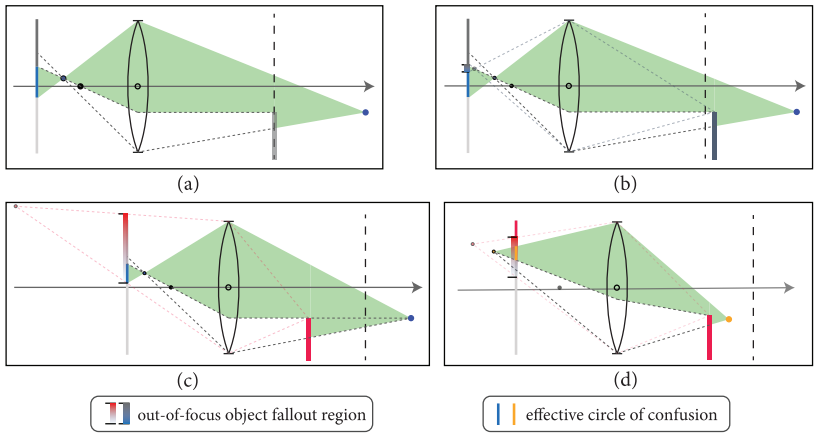


Figure 5.3: All scenarios where occulusion between objects at different depth induce changes to the CoC of the occuluded object. (a) occuluder at plane of focus; (b) occuluder further than plane of focus; (c) object closer than plane of focus occuluding object further than plane of focus; (d) object closer than plane of focus occuluding another object closer than plane of focus.

depth of the plane of focus z_f into consideration: $z_f < z_2 < z_1$; $z_f = z_2 < z_1$; $z_2 < z_f < z_1$; $z_2 < z_f = z_1$; and $z_2 < z_1 < z_f$. Given the fact that it is the CoC of the occuluded object that we are interested in (Figure 5.2 (d)), we could eliminate the case where $z_f = z_1$, as in this case the partially occuluded object will always be perfectly in focus with zero sized CoC, albeit darkened. All other four scenarios are illustrated in Figure 5.3.

DOF MODEL IN RENDERING In the context of rendering with perspective camera models (e.g., pinhole camera), the optical model in Figure 5.1 is simplified [37]. Instead of explicitly putting a lens element in front of the imaging plane and imaging everthing upside down, we could opt for a simpler layout where the imaging plane is placed at the same side as the viewed scene. Points in scene are projected towards a finite sized disk instead of a single viewpoint. Due to the fact that perspective camera transformations do not include the non-linear imaging process as described in Equation 5.1, we can further substitute notions in Equation 5.2 and get $r = \frac{|z-z_f|}{z}R$. It can be shown that this simplified model converges to the same CoC size if we set the disk to have a radius of r_0, i.e., $r = \frac{|z-z_f|}{z}r_0$. Figure 5.5 illustrates such a camera model.

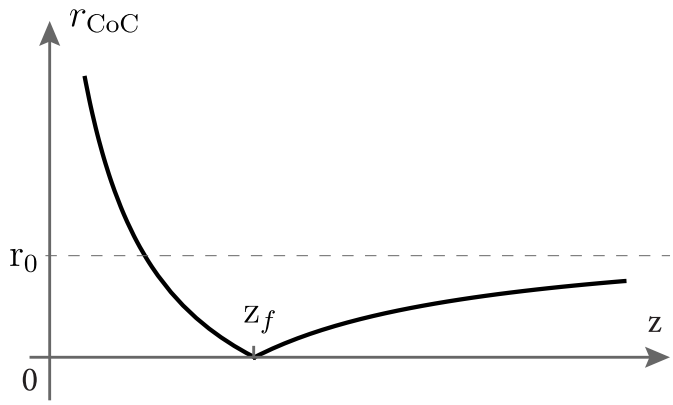


Figure 5.4: Plot of CoC size as a function of depth z (Equation 5.3). On the near side, notice how as z approaches 0 the radius of CoC goes to infinity. Given that most rendering algorithms employs near clipping, the CoC remains bound in practice. On the far side, please note how the size of the CoC approaches r_0.

To summarize, we use a lens with parametrized radius of r_0 and a plane of focus distance of z_f, we have the radius of the CoC, r_{CoC}, for a point at depth z:

$$r_{\text{CoC}}(z) = \frac{\left\| z - z_f \right\|}{z} r_0. \tag{5.3}$$

5.2.1 *Prior Work*

Most ray-tracing based solutions simulate the optical systems of such cameras using distributed ray tracing [4]. Testing light rays against geometric lenses gives high-quality results but too few samples lead to noisy images.

Other approaches include multiview accumulation [9], which renders the image from multiple viewpoints and accumulates the results to create the final image. The views are chosen such that the objects in the focal plane align and remain sharp, whereas out-of-focus objects misalign, hence, producing blur. While good results are possible, multiview accumulation requires many render passes, making it costly.

Efficient real-time methods often filter a shaded image, involving its depth buffer. Several options for fast convolution have been proposed for real-time performance: a simple 3x3 filter [15, 22], downsampling and separable Gaussian filtering [39, 50], summed-area table [45], or anisotropically-filtered mipmap interpolation [64].

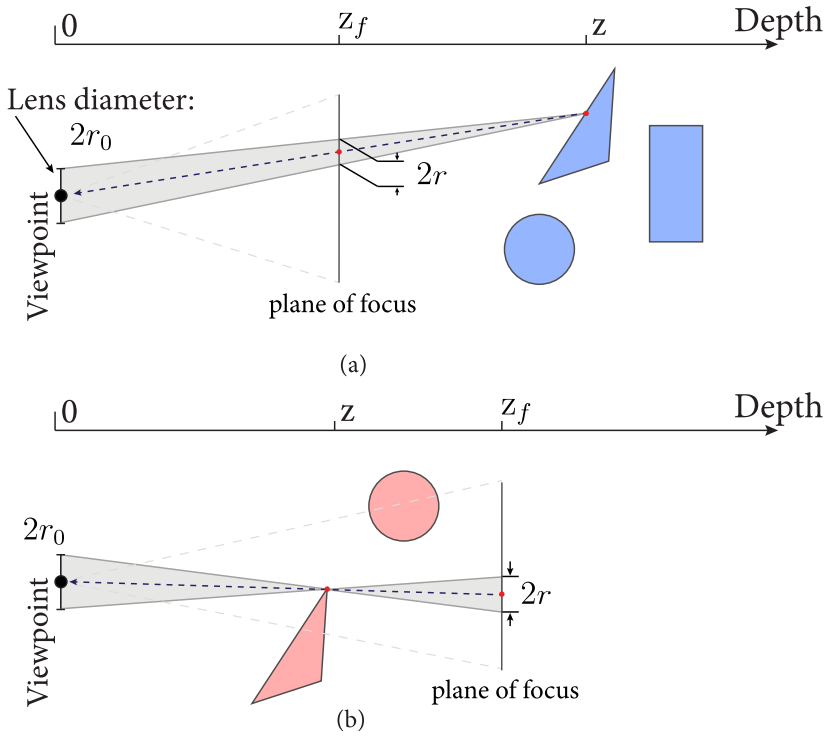


Figure 5.5: Simplified perspective camera illustration in 2D. Notice how the circle of confusion relates to the lens.

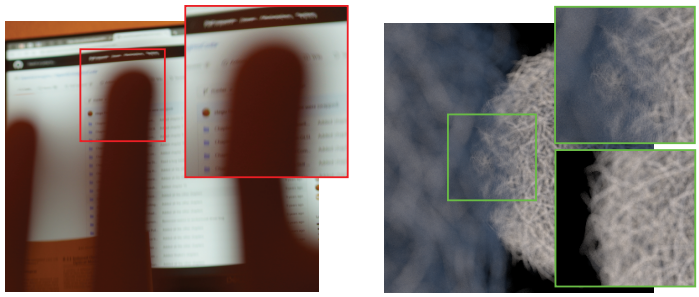
All methods above suffer from artifacts, such as *intensity leakage* that result from using a single image. Typically, these artifacts occur at *depth discontinuities*, where disocclusions would occur. The reason is a lack of information, which can only be coarsely approximated but not replaced by *anisotropic filtering* [41, 64], pyramidal image processing [52] or *gathering* [3, 62]. To recover the missing occluded information, multiple layers of the scene can be used, which often involves alpha blending. The decomposition can be limited to a chosen foreground/background [49] but more advanced postprocessing solutions discretize images into sub-images and filter them separately [43] or perform splatting [57]. Hybrid methods obtain a more accurate scene representations by extracting scene layers. Using ray tracing against this layered image-based representation results in near-correct results [63, 67]. Our approach avoids ray-tracing and opts for a more light-weight computation, while still capturing the complex visibility relations between the layers.

Most recent work focuses on the generation of the *bokeh* effect [80, 93], which is the result of out of focus light sources that produce a visible aperture shape in the image. Various techniques are proposed for fast computation: low-rank linear filtering [81] and polygonal shaped separable filtering [73]. Fast approximations using separable filter kernels [90] or precomputed lookup tables [104] can result in plausible approximations of this often artistic effect. More general artistic control over depth-of-field effects has been studied in various works: through placing lens points [13], semantic context based filtering [33], detail removal filtering instead of blurring [59], depth of field on non-realistically rendered skewed plane [40], user defined focus control [67], via user defined grayscale map to control a filter kernel [51] and modelling textured bokeh effects using a fractional Fourier transform [84]. Although this chapter focuses primarily on realistic rendering, it would be possible to offer several parameters for artistic control over the final output. Our method for realtime depth of field effects works by post-processing in screen space, resulting in a fast approximation of lensing and distortion phenomena.

5.3 OUR METHOD

In the inter-occlusions setting between objects, we specifically consider the case where objects on the camera's side of the plane of focus, referred to as *near objects*, partially occlude objects on the far side of the plane of focus, referred to as *far objects*, as in this situation the effect of lensing and distortion is most pronounced. Further, we refer to the view space closer than the plane of focus as *near field* and the farther space *far field*.

We first introduce the concept of the effective circle of confusion in Section 5.3.1 followed by an overview of the pipeline in Section 5.3.2.



(a) Real life phenomenon (b) Ray traced example

Figure 5.6: CoC lensing and distortion phenomenon in (a) real world and (b) ray traced images. (a) The author's fingers in front of a bright monitor: the focus lies somewhere between the fingers and the monitor. Notice how in the red-highlighted region what should have been blurred texts on the monitor now appear sharper and distorted around fingers. (b) A ray traced image of two hairballs, with the focal plane lying in between two balls. Notice in the green-highlighted region similar effects take place and the further hairball appears sharper and duplicated views appear.

Then explain how we handle model shrinking in Section 5.3.2.1 and distortion in Section 5.3.2.2, before continuing with the actual rendering alogrithm in Section 5.3.2. Finally, we explain how this model can be used to influence sampling strategies in offline rendering in Section 5.3.3.

5.3.1 *Effective Circle of Confusion*

Distortion and the lensing phenomena can be intuitively understood; near objects can block the light reaching the aperture of the camera lens or pupil/iris of eyes. This reduction makes an out-of-focus far object appear sharper, as shown in Figure 5.6 (a) (photo) and Figure 5.6 (b) (a ray-traced simulation). This phenomenon is actually made use of in some real-life situations; for instance, a pinhole occluder such as pinhole glasses are used by ophthalmologists, or people experiencing myopia tend to squeeze their eyes to reduce perceived blur.

To illustrate the phenomenon, please refer to the example as laid out in Figure 5.2 (d) and its simplified version in Figure 5.7. When projecting the far object (with depth z_1) towards the lens, a part of the lens is occluded by the near object (with depth z_2). As the occluded part will fail to make it to the lens, the footprint on the imaging plane does actually not resemble a circle. We name the corresponding actual footprint the *effective circle of confusion* (effective CoC).

Having the near object move along the grey arrow downwards as in Figure 5.7, more of the lens will be blocked for the far object by the near object, meaning that its effective CoC *shrinks*. Furthermore, as the CoC shape changes, the effective CoC's centre also appears to shift, causing a *distortion* effect (e.g. Figure 5.6 (a)). Notice that such effect is only perceived as such when viewed through the fallout regions of near object. In cases where the occuluder projects to a in-focus image, there is no blurred part to view through and far objects will simply appear overlaid.

Observing from Equation 5.3 and Figure 5.4, we could see that the size of CoC can be much larger on the near side of the plane of focus comparing to on the far side. As a result the fallout regions of near objects are much larger and therefore the phenomenon more easily observable and more pronounced. When occuluder resides in the far field (Figure 5.3 (b)), the fallout regions tend to be limited, plus the fact that the occuluded objects in this case would have a larger CoC due to depth difference, the phenomenon is much less observable. Therefore, we opt for simulating the lensing and distortion induced by near objects on far objects, as depicted in Figure 5.2 (d), Figure 5.3 (c) and the corresponding simplified model in Figure 5.7, i.e. $0 < z_2 < z_f < z_1$. In reality, any object not residing on the plane of focus can cause such a distortion and lensing effect, e.g., between two near objects when $0 < z_2 < z_1 < z_f$ and two far objects when $0 < z_f < z_2 < z_1$ (see Figure 5.3 (b) & (d)). The case as illustrated in Figure 5.3 (a) where the occuluder coincides the plane of focus indeed produces a smaller CoC. However, the lack of a fallout region through which the image of the occluded object is viewed means that this is a pure occulusion case without any lensing or distortion. Nevertheless, lensing and distortion effects in these cases are less pronounced. Their inclusion is left for future work.

5.3.2 *Pipeline Overview*

We work in screen space with filters. The depth dependent DoF effects are achieved with Gather based approach. Namely, we take as input the shaded image and its corresponding depth map. Typically with Gather approach, a pre-defined set of sample points is used. When scaled according to the CoC calculated using the depth value with Equation 5.3, the sample points will produce a filter with variable size. We affect this scaling step to achieve effective CoC.

When filtering in Gather, the center of the filter is defaulted to the pixel we aim to filter. In our case, however, we need to shift this centre according to a distortion direction. This acounts for the distrotion effect caused by effective CoC.

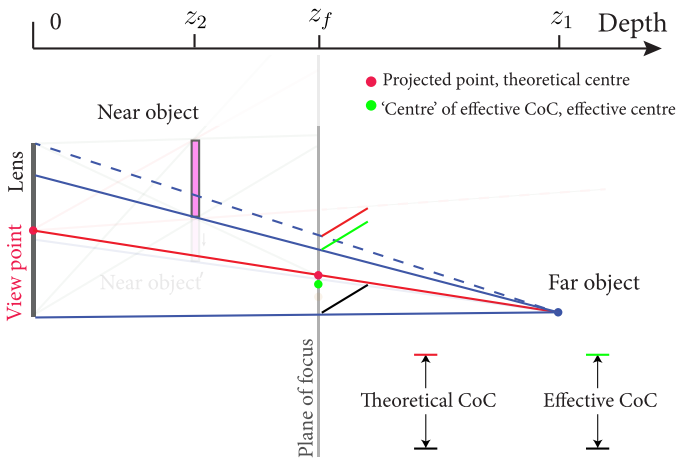


Figure 5.7: Illustration of *Effective CoC* with near object (in pink) obstructing the lens projection of an out-of-focus point in space (see also Figure 5.2 (d)). Notice how as the near object moves to a new position the effective CoC shrinks and its center moves accordingly.

In a rasterization rendering pipeline, the scene inside viewfrustrum is projected towards the viewpoint. As such, information that is occuluded would not be available for filtering. As the distortion effects include shifting filter kernels, this hidden information is crucial for the effects. Therefore, we opt for a layered rendering scheme.

We start by subdividing the viewfrustrum according to the near and far fields, which are respectively rendered to layers *near layer* and *far layer*. To comply with the geometric analysis when deriving the effective CoC size and shift (Equation 5.4 and Equation 5.5), we underline the necessity to have objects close to the plane of focus separately rendered as well. Therefore, apart from the near and far fields already defined, we add the additional *in-focus* fields and consequently *in-focus layer*. We render both the shaded image and depth of each field to separate layers.

When filtering, the near layer is filtered first, creating an filtered shaded image with alpha masked blur. The in-focus layer and the far layer are then subsequently added to the filtered near layer. When compositing pixels through the fallout regions of near layer, the alpha value and a vector that is maintained during filtering is retrieved. The in-focus layer and far layer are then composited in order, with the depth-dependent CoC calculated further scaled according to the alpha value in the filtered near layer, and filter kernel shifted according to the vector. See Figure 5.8 for an overview.

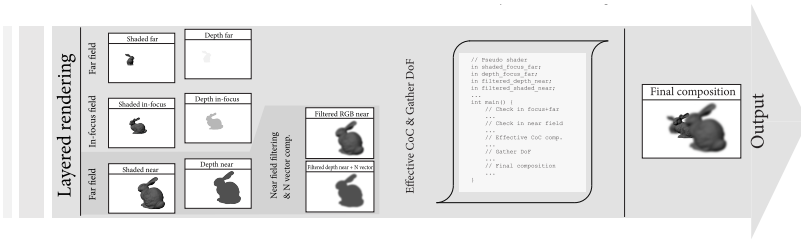


Figure 5.8: Overview of applying effective CoC in the context of real-time depth of field rendering. We start with three layers and filter the near layer to obtain filtered RGBA, filtered depth and shift vector.

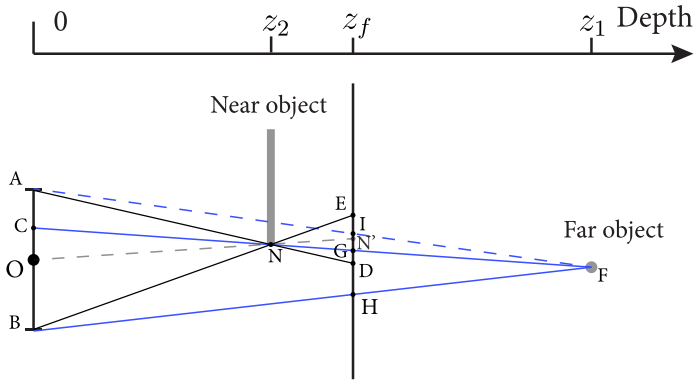


Figure 5.9: 2D illustration of Effective CoC with $0 < z_2 < z_f < z_1$. Notice the similar triangle pairs: $\triangle ABN$ and $\triangle DEN$, $\triangle ACN$ and $\triangle DGN$, and $\triangle FAC$ and $\triangle FIG$.

5.3.2.1 *Shrinking*

To model shrinking, we need to quantitatively describe the size of the effective CoC r_e. With reference to Figure 5.9 and Equation 5.3, we can project the far object F towards the lens AB through the plane of focus, leaving the footprint of IH. Near object N is projected likewise, resulting in ED. From here, one can derive:

$$r_{\text{CoC}} = IH = \frac{\left\| z - z_f \right\|}{z} r_0,$$
$$r_e = GH = r_{\text{CoC}} \cdot \frac{BC}{AB},$$

where IH, GH, BC and AB are triangle edge lengths that correspond to the size of: original CoC, eCoC, actual visible lens to object and original lens respectively.

Instead of solving for BC, which would require explicit ray casting with a significant amount of samples, we observe the similar triangle pair $\triangle ABN$ and $\triangle DEN$ and make use of the fact that $\frac{BC}{AB} = \frac{EG}{DE}$.

ED reflects the circle of confusion for point N, it would thus correspond to the filter of the near object N with centre being N' (the pinhole projection of N onto the image plane, i.e. projecting $\overrightarrow{ON}$). Our assumptions to only consider near objects inducing distortions on far objects implies that all near objects can be filtered independently from the far objects.

When near objects are filtered, their boundaries will have a gradual fall-off from opaque to transparent. Therefore, we get $\frac{EG}{DE} = 1 - \alpha(G)$, where $\alpha(G)$ denotes the opacity of *filtered* projection of near objects. We can render the near objects and process that image to derive the opacity. Alternatively, this layer of near objects can also be derived from a single rendered image involving the depth buffer.

In order to model an effective CoC for the far objects, we use the filtered layer of near objects. The effective CoC size r_e is then given as:

$$r_e = \frac{\left\| z - z_f \right\|}{z} \cdot [1 - \alpha(G)] \cdot r_0. \tag{5.4}$$

5.3.2.2 *Distortion*

The distortion effect, relates to a shift of the center of the filter region with respect to the original CoC. To this extent, during the filtering of the near layer, we derive an offset vector $\overrightarrow{N}$ that we will use to shift the filtered lookup:

$$\overrightarrow{N} = \frac{\sum_{i=1}^{M} \overrightarrow{v_i}\delta(p_i)}{\left\| \sum_{i=1}^{M} \overrightarrow{v_i}\delta(p_i) \right\|}, \tag{5.5}$$

where M denotes the number of samples used for filtering inside the filter window, $\overrightarrow{v_i}$ denotes the directional offset vector for sample $p_i \in R^2$ and $\delta(p_i)$ denotes the binary value indicating whether there is a valid value in sample p_i. The input of this filtering can be either the depth or the shaded image of the near field. Notice that $\overrightarrow{N}$ is normalized. Following Figure 5.10 and Figure 5.7, the effective CoC can maximally shift to the center of GH. Using this observation, we drive the distortion by scaling the vector $\overrightarrow{N}$ by $r_e\alpha$, which gives the shift a linear transition from 0 to its maximum extent r_e.

5.3.3 *Importance Function for Lens/Aperture Sampling*

The observations in Section 5.3, allows us to approximately predict how objects partially block each other in the presence of depth of field.

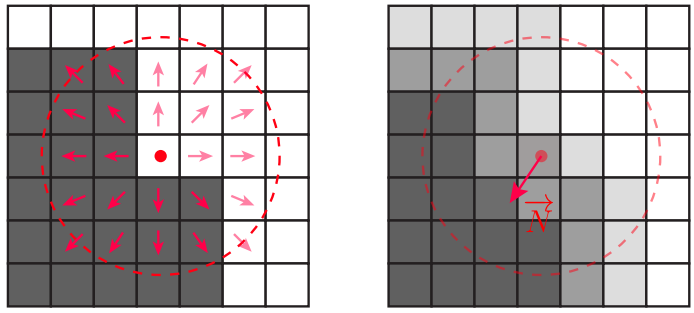


Figure 5.10: Illustration of maintaining the vector $\overrightarrow{N}$ during filtering process. The resulted vector $\overrightarrow{N}$ is the weighted average of all participating pixels in the filter window.

This information can be used to derive an importance function on the lens that can be used by a ray tracer to focus computations where they are most relevant.

In particle tracing algorithms, e.g. light tracing, spatial points carrying energy from light sources are projected towards the camera. In the case of a perspective camera with lens (see Figure 5.11), chances are that such splats are blocked by the very same configurations as discussed in Section 5.3. We propose an aperture/lens sampling strategy that takes as an input a depth map of the scene and outputs for each spatial query a lens sample with a high chance of splatting onto the image plane without being blocked.

With reference to Figure 5.9, we briefly describe idea of the algorithm and provide pseudocode in 5. For a spatial point F, we want to have the lens sample somewhere on BC over AC. Although technically in this illustration BC has more chance of being sampled than AC due to its larger area, we want to attribute more density to BC than to AC because AC is known to be blocked for F. We achieve this purpose by building a distribution over the lens, indicating the likelihood of mutual visibility between the sample point and the spatial query. Doing this naïvely would result in explicit tracing of many sample rays, which leads to a paradoxical situation. Therefore, we choose to subdivide the lens into equal areas and only test spatial query distance against pre-computed depth maps.

5.4 IMPLEMENTATIONS AND RESULTS

We implemented the applications above respectively. We build our layered rendering in Vulkan. We use reference OpenGL implementation from [63] as a ray traced solution for comparison. Additionally, we also

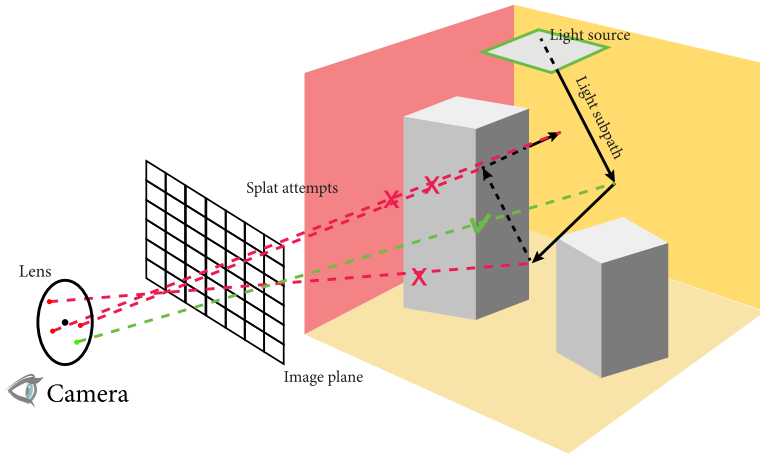


Figure 5.11: Illustration of blocked lens samples when splatting towards the camera lens.

Algorithm 5 Pseudo-code for aperture/lens sampling in realistic rendering

```
1: depthMap[S] // S: number of preset samples
2: lensArea[S]
3: procedure SampleLens(p)
4:     val ← ∅
5:     for each preset lens sample s_i do
6:         val[i] ← depthMap[s_i] >= ||p, s_i||
7:     cdf ← BuildCDF(val)
8:     ξ ← Rand2D()
9:     idx, pdf ← cdf.Sample(ξ.x)
10:    return lensArea[idx].Lerp(Remap(ξ.x), ξ.y), pdf
```

compare our method against state of the art non-ray tracing depth of field solutions, namely, one-pass and gather. Lens/aperture sampling is implemented in pbrt-v3 [85].

5.4.1 *Effective CoC for Real-time DoF with Layered Rendering*

With reference to Figure 5.12 in Section 5.3.2, we divide the view frustum to three layers: near, in-focus and far. All three layers are rendered to RGBA and depth.

Render layers are set according to depth (with reference to Figure 5.12). Given that the original view frustum ranges from z_0 to z_1. The new frustums are set to be:

- *near*: $z'_0 (= z_0)$ to z'_1;
- *in-focus*: z'_1 to z'_2;
- *far*: z'_2 to $z'_3 (= z_1)$.

Due to the fact that the CoC size within near field is typically large and that the difference within near field is barely noticeable, we choose to cap CoC size at a maximum R_{max} that is parameterized by z'_1 (Equation 5.3). We set the in-focus field and the far field to meet at the focus depth. The un-distorted CoC size is also plotted in Figure 5.12, with the only difference being the capped CoC size of near field at R_{max}.

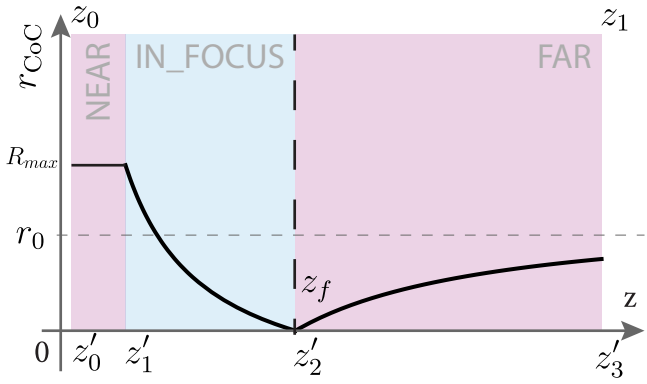


Figure 5.12: Illustration of three render layers: near, in-focus and far. Notice that in-focus and far meet at the focal depth while CoC radius in near is cap at a maximum radius. A small overlap between adjacent layers is introduced to avoid artifact durin compositing.

5.4.1.1 *Separated Near Field Filtering*

Since we are working with a relatively large window size for the near layer, a filter that can be separated is desirable. We opt for treating all participating terms independently, i.e., RGBA, depth, x and y components of vector $\overrightarrow{N}$.

We chose to use gaussian filtering for RGBA, as it is separable and produces more realistic blur comparing to box filter. For depth filtering, we perform a nearest like flooding operation that expands the boundary of near layer. This is used as depth reference for fallout regions of near field during final composition.

The vector $\overrightarrow{N}$ is a weighted average of offsets of all contributing pixels as noted in Equation 5.5: This operation, however, uses different weights for horizontal and vertical passes, that is for the x component:

$$\overrightarrow{N}.x = \sum \overrightarrow{v_i}.x \cdot \delta(p_i) = \sum^{Y}\sum^{X} (x_i - x_0) \cdot \delta(p_i),$$

which treats the vertical direction equally but uses signed offset in the horizontal direction as weights. The y component was separated in the opposite order. Notice that the normalization operation is not included here as the vector is normalized during final composition and is omitted here.

The resulted shader code for near layer horizontal filtering is given in the following Listing 1. The vertical pass omitted due to similarity.

5.4.1.2 *Final Composition*

The final compositing pass largely follows the process as discussed in Section 5.3.2. A modified gather operation with variable maximum radius is used with 48 samples that are distributed in 3 scaled circles around the center [93]. A view decomposition is given in Figure 5.13.

We implemented commonly used methods: one pass, gather DoF and our method in Vulkan. We use the implementation from state of the art ray traced DoF [104]. Due to the proprietary design used in their framework, the handling of scene, camera transformation and the shading are not open to us and the overall appearance of renders are not entirely the same. Nevertheless, we aim to compare the lensing and distortion of out-of-focus objects that are present.

We rendered two scenes for this experiment, the *Sponza* scene and the *Bistro interior* scene. Both scenes feature complex geometry with many object inter-occlusions. The resutls are presented in Figure 5.14 and Figure 5.15. Most leave geometries in *Sponza* use alpha masked texture, which shows the compatibility of our method in this regard. The runtime performance comparison of all experiments at FullHD resolution is given in Table 5.1.

```
#version 450
layout( set=0, binding=0 )
uniform sampler2D uTexColor;
layout( set=0, binding=1 )
uniform sampler2D uTexDepth;
layout( location = 0 ) in vec2 v2fTexCoord;
layout( location = 0 ) out vec4 oColorX;
layout( location = 1 ) out vec2 oMagicX;
layout( location = 2 ) out float oDepthX;
void main()
{
  int sigma = MAX_COC_RADIUS/3;
  vec2 pixelSize =
  vec2(1.f/textureSize(uTexColor, 0 ).r);
  int bx = -3 * sigma;
  int ex = 3 * sigma + 1;
  vec4 resRGBx = vec4(0.f);
  vec2 resMagx = vec2(0.f);
  float resDepth = 1.f;
  int i = 0, j = bx;
  for (int x = bx; x < ex; ++x)
  {
    vec2 offset = vec2(x, 0);
    vec2 tc = v2fTexCoord+offset*pixelSize;
    vec4 spl = texture(uTexColor, tc);
    resRGBx += spl * kGausblurCoeff_sig5[i];
    resMagx.x += float(j) * float(spl.w);
    resMagx.y += float(spl.w);
    ++i; ++j;
    resDepth = min(resDepth,
    texture(uTexDepth, tc).r);
  }
  oColorX = resRGBx;
  oMagicX = resMagx;
  oDepthX = resDepth;
}
```

Listing 1: Horizontal pass shader code for near layer

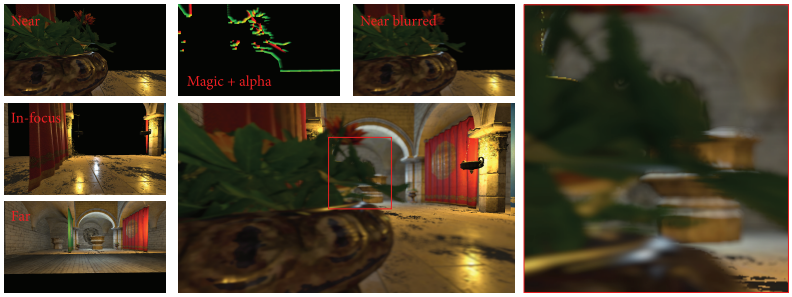


Figure 5.13: The decomposition of rendering the *Sponza* scene. The near blur buffer and the magic+alpha buffer are obtained from near layer filtering using our separated filtering. The final composition is merged result by adapting gather kernel radius from the in-focus layer and the far layer. A close-up of the lensing and distortion effect is highlighted in the inset.

As can be clearly seen from the results, our method effectively models the out-of-focus objects lensing and distortion effects. Due to the nature of gather DoF handling blending, the near objects suffer from boundary issue that is constantly observed across scenes. While ray traced results appear more accurate, our method runs at much higher frame rate (ranging from 3x to 300x) when the lensing and distortion effects are desired. Being a screen space technique, our method enjoys very stable performance behavior, which is of high importance in real-life scenarios.

5.4.2 *Aperture Sampling*

For aperture samping, we opt for a preset of 9 samples, spanning $(0, 1]^2$ that are eventually mapped to a disk of size r using concentric disk mapping [27]. Due to the nature of the mapping, we are able to divide the disk into 9 sub areas with equal area, which simplifies the subsequent sampling w.r.t. density attribution.

As described in 5, we precompute 9 depth maps with higher resolution. In the context of offline rendering solutions, rendering one frame of single depth ray casting images takes negligible time (avg. 0.3s for *Kitchen*, 0.8s for *Classroom* and 1.1s for *Staircase*) comparing to the whole offline render. The rest of the implementation are done exactly as 5. Visual results are plotted in Figure 5.16, Figure 5.17 Figure 5.18. We report other important statistics in Table 5.2.

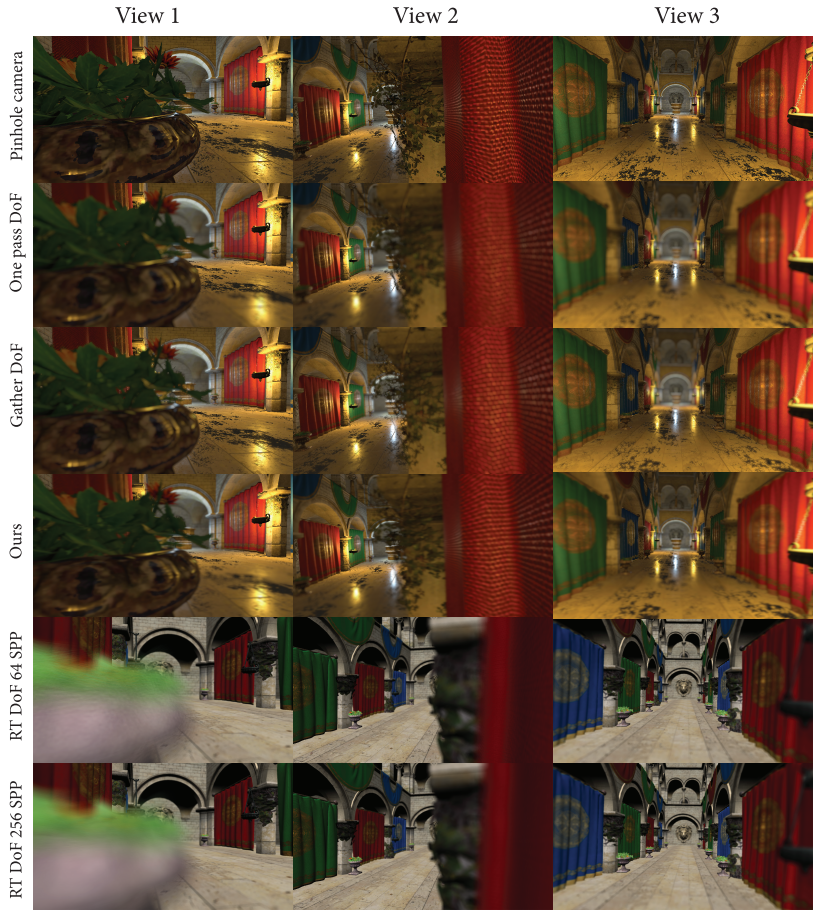


Figure 5.14: Views of the *Sponza* scene. All tests run at FullHD. See Table 5.1 for runtime performance breakdown. Despite the different appearance due to internal shading difference, the lensing and distrotion effect is pronounced in both our method and ray traced results (RT 64 samples and RT 256 samples). Ray traced DoF achieves high framerate in view 3 due to the fact that in this view the near object occlusions is close to none.

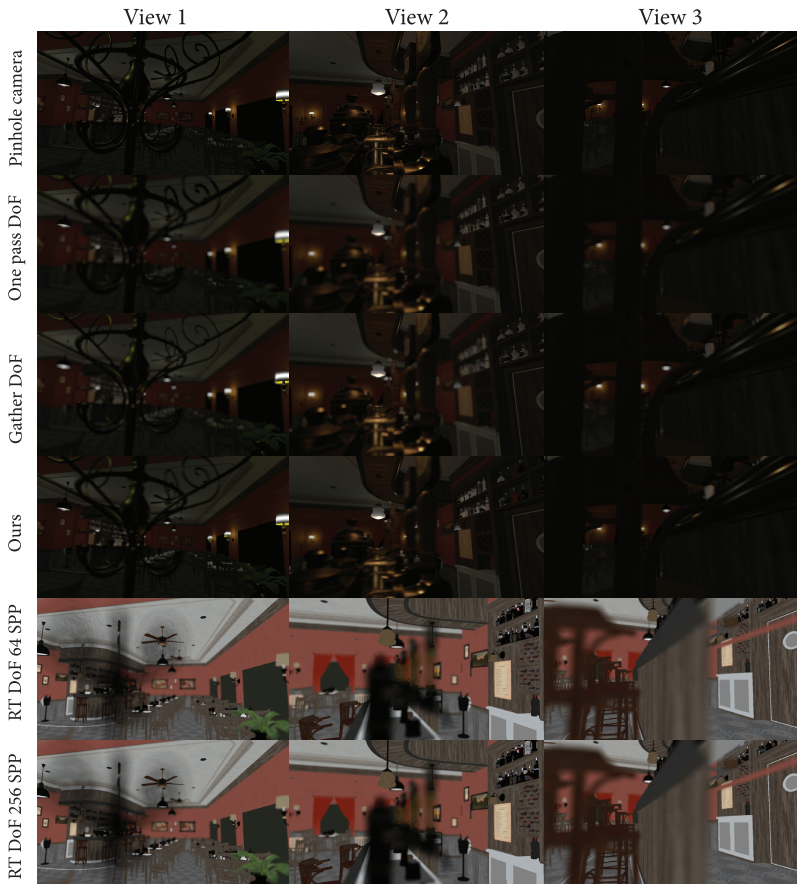


Figure 5.15: Views of the *Bistro interior* scene. All tests run at FullHD. See Table 5.1 for runtime performance breakdown.

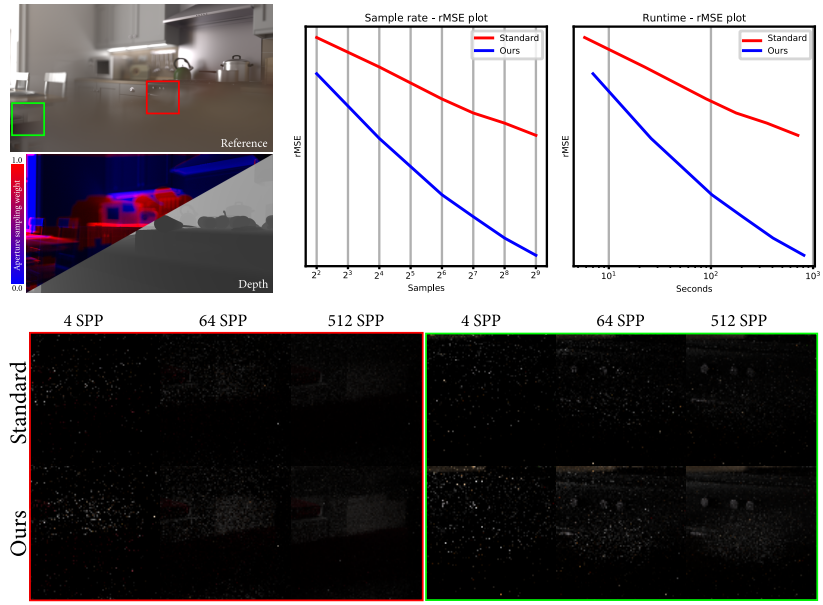


Figure 5.16: Results for testing our aperture sampling in the modified *Kitchen* scene. Since we are doing light tracing, images took much longer to converge than traditional tracing methods from the camera. Nevertheless, we benifit from our sampling and manages to converge much faster, both in terms of sample rate and in terms of runtime.

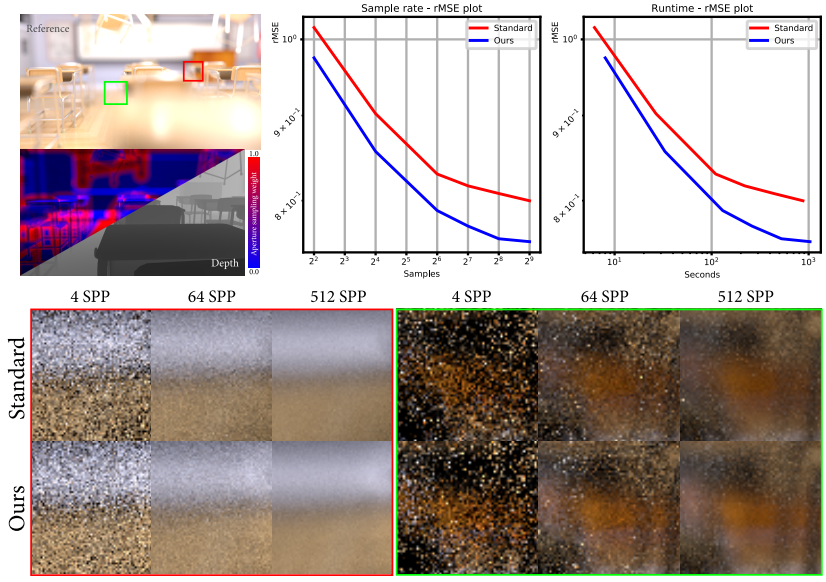


Figure 5.17: Results for testing our aperture sampling in the modified *Classroom* scene. This scene features many light sources and contributes to faster convergence comparing to Figure 5.16.

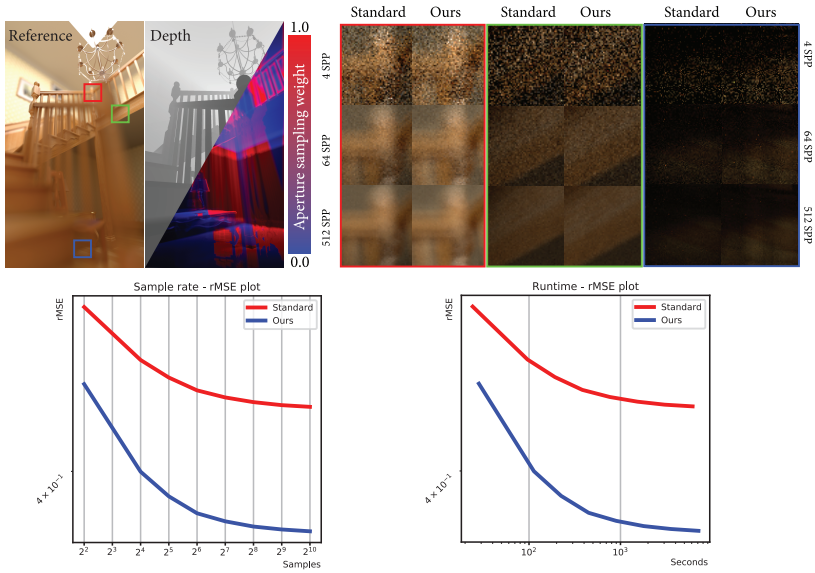


Figure 5.18: Results for testing our aperture sampling in the modified *Staircase* scene. Notice aperture sampling results in less black pixels and produces smooth results faster.

		Sponza			Bistro interior		
		View 1	View 2	View 3	View 1	View 2	View 3
One pass	total	1030μs	980μs	940μs	1560μs	1880μs	1820μs
	render	470μs	420μs	360μs	980μs	1320μs	1260μs
	filtering	560μs	560μs	580μs	580μs	560μs	560μs
Gather	total	760μs	724μs	632μs	1321μs	1562μs	1573μs
	render	460μs	410μs	350μs	950μs	1250μs	1250μs
	coc	30μs	30μs	28μs	28μs	28μs	30μs
	downsample	90μs	90μs	94μs	90μs	90μs	90μs
	nearest	30μs	27μs	26μs	27μs	26μs	27μs
	blur	20μs	20μs	20μs	20μs	20μs	20μs
	compute	100μs	114μs	86μs	170μs	114μs	124μs
	fill	30μs	33μs	28μs	36μs	34μs	33μs
	composite	90μs	90μs	92μs	87μs	88μs	89μs
Ours	total	1920μs	1940μs	1880μs	2760μs	3340μs	3030μs
	render	700μs	680μs	510μs	1350μs	1870μs	1650μs
	nearX+Y	1010μs	1030μs	1000μs	1030μs	1060μs	1030μs
	merge	210μs	230μs	370μs	380μs	410μs	350μs
RT 64	total	140ms	135ms	6ms	43ms	16ms	15ms
RT 256	total	575ms	540ms	14ms	165ms	54ms	48ms

Table 5.1: Runtime performance measurements and breakdowns of Figure 5.14 and Figure 5.15. Notice the unit difference: μs and ms.

		Runtime (in seconds)						Average blocked splat
		4 SPP	16 SPP	64 SPP	128 SPP	256 SPP	512 SPP	
Kitchen	Standard	5.8	22.4	90.1	176.4	353.2	710	61.9%
	Ours	7	25.9	102	203.6	405.2	812.9	43.7%
Classroom	Standard	6.2	26.7	110.1	219.3	436.3	874.0	60.9%
	Ours	8.0	33.0	130.0	260.7	524.2	1033.4	42.2%
Staircase	Standard	24.2	96.3	382.7	763.4	1532	3040.8	59.9%
	Ours	28.3	113.1	448.4	897.6	1792.8	3561.9	44.4%

Table 5.2: Runtime and averaged blocked splat ratios for tests in Figure 5.16, Figure 5.17 and Figure 5.18.

As can be seen from the results, by making use of depth information, we could derive a better sampling distribution that yields higher successful splats. Overall, we managed to improve by around 40.5% unblocked splats. As a result, both at equal sample rate and with equal runtime, our method gives more non-black pixels, which are evidently shown in Figure 5.16 and Figure 5.17.

5.5 CONCLUSIONS

In this chapter we presented an extended lens model for perspective cameras. We modeled circle of confusion shrinking and distortion effects caused by inter object occlusions. Insights from analysis of the geometrical optics behind perspective camera lens enable better lens sampling in particle tracing.

6 CONCLUSION

To infinity and beyond.

— Buzz Lightyear

In this thesis, we have proposed multiple methods for improving the efficiency of realistic rendering for both offline and real-time solutions. For offline rendering, we have focused on the light transport simulation and treated the sampling phase and the reconstruction phase. In real-time rendering, we addressed a visual phenomenon through modeling.

Throughout the chapters, we have made use of pre-computations. This is the case in Chapter 2 and Chapter 3. We have shown how to design data structures to collect and maintain specific information and demonstrated the benefits of incorporating this knowledge during rendering.

In Part i, we break down the path construction process. We proposed primary sample space path guiding in Chapter 2. As the target is to avoid caching spatially, we designed the adaptive k-dimensional binary tree and the accompanying traversal methods for path radiance information. The guided path tracing compared favourably against its counterparts and higher efficiency is achieved by delivering lower variance results with equal sample rate. A spatial to spatial mapping structure is developed in Chapter 3 to assist visibility information collection and query. We have shown the gain of making use of prior visibility information with the demonstration of multiple applications in the context of path tracing and bi-directional path tracing. Better informed path connections and significantly reduced cost on shadow ray tests resulted in lower variance and faster convergence.

The unbiased approaches as presented in Part i towards the sampling phase have shown benefit of spending some runtime on collecting scene information prior to the actual path tracing. Despite the advent of high performance graphics computing devices tailored to ray tracing, multi-pass algorithmic designs prove effective in the off-line rendering context that is dominated by numerical integration solutions.

The main reason behind the success of this scheme is that deriving prior knowledge, we not only have a rough idea of where samples should be placed, but also how associated function evaluations should be weighted.

Apart from the classic approach of weighting samples with a probability density, we also investigated the possibility of associating sample weights with neighbourhood information during reconstruction in Part ii. We revisited the very basics of Monte Carlo integration and pro-

posed geometric sample reweighting that made use of sample adjacency property while maintaining unbiasedness in Chapter 4.

We have shown how the classic Monte Carlo integration can be reformulated in the context of a fixed sample set size. Naively deriving weights from the sample neighborhoods introduces a systemic bias. We demonstrated how this bias could be corrected, arriving at an unbiased solution. The performance is close to that of low discrepancy sequences is shown in both numerical experiments and rendering tests. To further expand the applicability of our reweighting scheme, we also developed a technique to work with piece-wise constant non-uniform samples that are commonly seen in rendering problems.

This different perspective of looking at Monte Carlo integration provided us with new possibilities for variance reduction. Since the bottleneck of many problems comes from integrand evaluation, limiting the number of samples is a common constraint when performance is crucial. Improving the numerical performance for a fixed sample set size is therefore a major objective. Before, this can be achieved with carefully chosen quasi random numbers for improved variance performance. With the technique we presented, we can loosen the requirement for input samples and still enjoy improved yet unbiased results.

In Part iii we have addressed a real-time rendering problem that previously only has ray traced solution for full effect. The depth of field effects caused by out-of-focus blur is natively solved with ray tracing in offline rendering. In the real-time rendering domain, we have shown that even state-of-the-art ray tracing approaches require too many samples to meet a real-time target. We have investigated the optics behind the phenomenon and proposed the effective circle of confusion to model distortion effects that were previously neglected in Chapter 5. At the cost of merely a hundredth of ray tracing, the easy-to-implement screen-space technique delivered a plausible approximation.

In general, the new methods presented in this thesis aim to improve efficiency of rendering pipelines. The methodology, however, could be applied beyond the rendering context. Multi-pass methods with pre-computing will, in the foreseeable future, enjoy wide adoption despite the many advantages of on-the-fly computing. Path guiding in primary sample space shares some resemblance with path finding routines in the robotics domain, where local decisions are dominating the finding process. The scheme in visibility mapping of voxelizing the scene could find application for other mutual connectivity information than visibility in domains such as urban planning. These information could range from road to network connection, from noise to air flow throughput. The efficiency by simply making use of neighbourhood adjacency property with geometric sample reweighting can be immediately applied in virtually any other field where Monte Carlo integration is used. The

effect of circle of confusion distortion is orthogonal to many integrating effects where a defacto observer with surface is involved. The closest example is the interference of soft shadows when objects at different distances to light source are occluding each other. Moreover, image processing tasks can trivially make use of the model out of box.

Looking back at the work carried out in this thesis, it is safe to say the rendering research has been extended by the few problems that we answered and, at the same time, by the questions that we raised explicitly or implicitly. We touched from the very underling mathematics foundations to the new high performance models along the path of approaching realistic rendering, which makes this thesis unique.

Looking outside the graphics community, we are humbled by the advances in peer communities such as computer vision and learning based artificial intelligence. It is safe to claim that the advances of graphics hardware is no longer the sole demand of gamers or the graphics community, but also that of communities that make heavy use of machine learning and deep learning. Speaking from a graphics researcher point of view, we want to raise the question as to whether it is enough to have weak models and plenty of training data to ultimately and successfully model or solve a real world problem with machine learning or deep learning techniques. Although the definition of solving a problem varies from case to case, and most of the time real world scenarios require 'good-enough' solutions per case instead of generic solution, we want to emphasize on the fact that there are numerous problems that we humans with current knowledge can not even describe, let alone design a loss function. We would argue that strong model approach will find its place, sooner than later, not necessarily to replace but to assist current data-driven approach.

Finally, it is our sincere hope that the methods proposed in this thesis will contribute to making ultimate realism for interactive photorealistic immersion a reality in the future.

APPENDIX

A

APPENDIX

A.1 EXPECTATION OF INTEGRATION OVER GIVEN SAMPLE SET

Here, we demonstrate the process of deriving expectation of a classic MC estimator. We make use of the following identity, common for multiple integration:

$$\int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1 = \int_0^1 \int_0^{x_N} \cdots \int_0^{x_2} \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_N.$$

It can be shown by induction that:

$$\int_0^{x_{i+1}} \int_0^{x_i} \cdots \int_0^{x_2} \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_i = \frac{x_{i+1}^i}{i!},$$
$$\int_{x_{i-1}}^1 \int_{x_i}^1 \cdots \int_{x_{N-1}}^1 \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_i = \frac{(1-x_{i-1})^{N-i+1}}{(N-i+1)!}.$$

Plugging these results into Equation 4.5 gives:

$$\begin{aligned}
\mathbb{E}\left[\hat{I}_{MC}\right] &= N! \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \\
&\quad \left[\frac{1}{N} f(x_1) + \frac{1}{N} f(x_2) + \cdots + \frac{1}{N} f(x_N)\right] \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1 \\
&= N! \cdot \frac{1}{N} \left[\int_0^1 f(x_1) \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \mathrm{d}x_N \cdots \mathrm{d}x_2 \mathrm{d}x_1 + \cdots \right. \\
&\quad + \int_0^1 \cdots \int_{x_{i-1}}^1 f(x_i) \cdots \int_{x_{N-1}}^1 \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1 + \cdots \\
&\quad \left. + \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 f(x_N) \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1 \right] \\
&= N! \cdot \frac{1}{N} \left[\int_0^1 \frac{(1-x_1)^{N-1}}{(N-1)!} f(x_1) dx_1 + \cdots \right. \\
&\quad + \int_{x_{i-1}}^1 \frac{x_i^{i-1}}{(i-1)!} \frac{(1-x_i)^{N-i}}{(N-i)!} f(x_i) \mathrm{d}x_i + \cdots \\
&\quad \left. + \int_{x_{N-1}}^1 \frac{x_N^{N-1}}{(N-1)!} f(x_N) \mathrm{d}x_N \right]
\end{aligned}$$

$$
\begin{aligned}
&= N! \cdot \frac{1}{N} \left[\int_0^1 \frac{(1-x)^{N-1}}{(N-1)!} f(x) \mathrm{d}x + \cdots + \right. \\
&\qquad \int_0^1 \frac{x^{i-1}}{(i-1)!} \frac{(1-x)^{N-i}}{(N-i)!} f(x) \mathrm{d}x + \cdots \\
&\qquad \left. + \int_0^1 \frac{x^{N-1}}{(N-1)!} f(x) \mathrm{d}x \right] \\
&= N! \cdot \frac{1}{N} \int_0^1 \sum_{i=1}^{N} \frac{x^{i-1}}{(i-1)!} \frac{(1-x)^{N-i}}{(N-i)!} f(x) \mathrm{d}x \\
&= \int_0^1 \sum_{i=1}^{N} \frac{(N-1)!}{(i-1)!(N-i)!} x^{i-1} (1-x)^{N-i} f(x) \mathrm{d}x \\
&= \int_0^1 \cdot [x + (1-x)]^{N-1} \cdot f(x) \mathrm{d}x = \int_0^1 f(x) \mathrm{d}x.
\end{aligned}
$$

A.2 EXPECTATION OF INTEGRATION WITH AUGMENTED PERIODIC SAMPLES

By augmenting samples to get a periodic sample set (one can imagine this process as creating a circular domain, where the last and first sample are hereby influencing each other's weight):

$$
\begin{aligned}
&\mathbb{E}\left[\hat{I}_P\right] = \\
&\quad N! \int_0^1 \int_{x_1}^1 \cdots \int_{x_{N-1}}^1 \\
&\qquad \left[\frac{1 + x_2 - x_N}{2} f(x_1) + \sum_{i=2}^{N-1} \frac{x_{i+1} - x_{i-1}}{2} f(x_i) + \right. \\
&\qquad \left. \left(\frac{1 + x_1 - x_{N-1}}{2} \right) f(x_N) \right] \mathrm{d}x_N \mathrm{d}x_{N-1} \cdots \mathrm{d}x_1,
\end{aligned}
$$

which integrates to $\int_0^1 ((-1)^{1+n} n(x-1)^n (n+x-2) + (x-1)(nx + 2x - 1 - (1-x)^n (n+x-1+(1/(1-x))^n (nx+1))))/(2(x-1)^2) f(x) \mathrm{d}x$. Again, this process results in a biased estimator.

BIBLIOGRAPHY

[1] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching." In: *Communications of the ACM* 18.9 (1975), pp. 509–517. DOI: 10.1145/361002.361007.

[2] S Yakowitz, JE Krimmel, and F Szidarovszky. "Weighted monte carlo integration." In: *SIAM Journal on Numerical Analysis* 15.6 (1978), pp. 1289–1300.

[3] M. Potmesil and I. Chakravarty. "A lens and aperture camera model for synthetic image generation." In: *Proc. ACM SIGGRAPH* 15.3 (1981), pp. 297–305. DOI: 10.1145/800224.806818.

[4] Robert L. Cook, Thomas Porter, and Loren Carpenter. "Distributed Ray Tracing." In: *Computer Graphics* 18.3 (1984), pp. 137–145. DOI: 10.1145/964965.808590.

[5] Robert L Cook, Thomas Porter, and Loren Carpenter. "Distributed ray tracing." In: *ACM SIGGRAPH Computer Graphics*. Vol. 18. 3. ACM. 1984, pp. 137–145. DOI: 10.1145/800031.808590.

[6] Jeffrey S. Vitter. "Random Sampling with a Reservoir." In: *ACM Trans. Math. Softw.* 11.1 (1985), pp. 37–57. DOI: 10.1145/3147.3165.

[7] James T. Kajiya. "The Rendering Equation." In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. ACM, 1986, pp. 143–150. DOI: 10.1145/15922.15902.

[8] James Arvo and David Kirk. "Particle transport and image synthesis." In: *ACM SIGGRAPH Computer Graphics* 24.4 (1990), pp. 63–66. DOI: 10.1145/97879.97886.

[9] P. Haeberli and K. Akeley. "The Accumulation Buffer: Hardware Support for High-Quality Rendering." In: *Proc. ACM SIGGRAPH* (1990), pp. 309–318. DOI: 10.1145/97879.97913.

[10] Don P Mitchell. "The antialiasing problem in ray tracing." In: *Advanced Topics in Ray Tracing, SIGGRAPH 1990 Course Notes* (1990).

[11] Franz Aurenhammer. "Voronoi diagrams—a survey of a fundamental geometric data structure." In: *ACM Computing Surveys (CSUR)* 23.3 (1991), pp. 345–405. DOI: 10.1145/116873.116880.

[12] Peter S Shirley. "Physically based lighting calculations for computer graphics." PhD thesis. University of Illinois at Urbana-Champaign, 1991.

[13] Michael Gleicher and Andrew P. Witkin. "Through-the-lens camera control." In: *SIGGRAPH*. 1992, pp. 331–340. DOI: 10.1145/133994.134088.

[14] Gregory J Ward. "Measuring and modeling anisotropic reflection." In: *ACM SIGGRAPH Computer Graphics* 26.2 (1992), pp. 265–272. DOI: 10.1145/133994.134078.

[15] P. Rokita. "Fast Generation of Depth of Field Effects in Computer Graphics." In: *Computers and Graphics* 17.5 (1993), pp. 593–595.

[16] Pierre Tellier and Kadi Bouatouch. "Physics-Based Lighting Models: Implementation Issues." In: *Photorealistic Rendering in Computer Graphics*. Springer, 1994, pp. 112–121. DOI: 10.1007/978-3-642-57963-9_12.

[17] Gregory J Ward. "Adaptive shadow testing for ray tracing." In: *Photorealistic Rendering in Computer Graphics*. Springer, 1994, pp. 11–20. DOI: 10.1007/978-3-642-57963-9_2.

[18] Henrik Wann Jensen and Niels Jørgen Christensen. "Photon maps in bidirectional Monte Carlo ray tracing of complex objects." In: *Computers & Graphics* 19.2 (1995), pp. 215–224. DOI: 10.1016/0097-8493(94)00145-0.

[19] Eric Veach and Leonidas J Guibas. "Optimally combining sampling techniques for Monte Carlo rendering." In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. 1995, pp. 419–428. DOI: 10.1145/218380.218498.

[20] Eric Veach and Leonidas Guibas. "Bidirectional estimators for light transport." In: *Photorealistic Rendering Techniques*. Springer, 1995, pp. 145–167. DOI: 10.1007/978-3-642-87825-1_11.

[21] Kurt Zimmerman and Peter Shirley. "A two-pass solution to the rendering equation with a source visibility preprocess." In: *Eurographics Workshop on Rendering Techniques*. Springer. 1995, pp. 284–295. DOI: 10.1007/978-3-7091-9430-0_27.

[22] Przemyslaw Rokita. "Generating Depth-of-Field Effects in Virtual Reality Applications." In: *IEEE Comp. Graph. and its App.* 16.2 (1996), pp. 18–21. DOI: 10.1109/38.486676.

[23] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. "Monte carlo techniques for direct lighting calculations." In: *ACM Transactions on Graphics (TOG)* 15.1 (1996), pp. 1–36.

[24] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. "Computational geometry." In: *Computational geometry*. Springer, 1997, pp. 1–17.

[25] Eric PF Lafortune, Sing-Choong Foo, Kenneth E Torrance, and Donald P Greenberg. "Non-linear approximation of reflectance functions." In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, pp. 117–126. DOI: 10.1145/258734.258801.

[26] Michael D McCool and Peter K Harwood. "Probability trees." In: *Graphics Interface*. Vol. 97. 1997, pp. 37–46.

[27] Peter Shirley and Kenneth Chiu. "A low distortion map between disk and square." In: *Journal of graphics tools* 2.3 (1997), pp. 45–52. DOI: 10.1080/10867651.1997.10487479.

[28] Eric Veach. "Robust monte carlo methods for light transport simulation." PhD thesis. Stanford University, 1997.

[29] Eric Veach and Leonidas J Guibas. "Metropolis light transport." In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, pp. 65–76. DOI: 10.1145/258734.258775.

[30] Russel E. Caflisch. "Monte Carlo and quasi-Monte Carlo methods." In: *Acta Numerica* 7 (1998), pp. 1–49.

[31] Eric Veach. "Robust Monte Carlo Methods for Light Transport Simulation." PhD thesis. Stanford University, 1998. URL: http://graphics.stanford.edu/papers/veach_thesis/.

[32] Eugene Hecht. "Optics 4th edition." In: *Optics 4th edition by Eugene Hecht Reading* (2001).

[33] Robert Kosara, Silvia Miksch, and Helwig Hauser. "Semantic Depth of Field." In: *Proc. IEEE Information Visualization*. 2001, pp. 97–104. URL: https://dl.acm.org/doi/10.5555/580582.857724.

[34] Sebastian Fernandez, Kavita Bala, and Donald P Greenberg. "Local Illumination Environments for Direct Lighting Acceleration." In: *Rendering Techniques* 2002 (2002), 13th. DOI: 10.5555/581896.581898.

[35] Heinrich Hey and Werner Purgathofer. "Importance sampling with hemispherical particle footprints." In: *Proceedings of the 18th spring conference on Computer graphics*. ACM. 2002, pp. 107–114.

[36] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. "A simple and robust mutation strategy for the metropolis light transport algorithm." In: *Computer Graphics Forum*. Vol. 21. 3. Wiley Online Library. 2002, pp. 531–540. DOI: 10.1111/1467-8659.t01-1-00703.

[37] Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2002.

[38] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. "Structured Importance Sampling of Environment Maps." In: SIGGRAPH '03. San Diego, California: Association for Computing Machinery, 2003, pp. 605–612. ISBN: 1581137095. DOI: 10.1145/1201775.882314.

[39] G. Riguer, N. Tatarchuk, and J. Isidoro. "Real-time depth of field simulation." In: *ShaderX*^{2}. Ed. by W. F. Engel. Wordware, 2003. Chap. 4, pp. 529–556.

[40] Brian A. Barsky and Egon Pasztor. "Rendering skewed plane of sharp focus and associated depth of field." In: *SIGGRAPH Sketches*. 2004, p. 92. DOI: 10.1145/1186223.1186338.

[41] Marcelo Bertalmío, Pere Fort, and Daniel Sánchez-Crespo. "Real-Time, Accurate Depth of Field using Anisotropic Diffusion and Programmable Graphics Cards." In: *Proc. 3DPVT*. 2004, pp. 767–773. DOI: 10.1109/TDPVT.2004.1335393.

[42] Herbert Aron David and Haikady Navada Nagaraja. *Order statistics*. 3rd. Wiley Online Library, 2004. ISBN: 9780471722168. DOI: 10.1002/0471722162.

[43] Brian Barsky, Michael Tobias, Derrick Chu, and Daniel Horn. "Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques." In: *Graphical Models* 67 (2005), pp. 584–599. DOI: 10.1016/j.gmod.2005.01.009.

[44] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. "Wavelet importance sampling: efficiently evaluating products of complex functions." In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 1166–1175. DOI: 10.1145/1186822.1073328.

[45] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. "Fast summed-area table generation and its applications." In: *Computer Graphics Forum* 24.3 (2005), pp. 547–556. DOI: 10.1111/j.1467-8659.2005.00880.x.

[46] Justin Talbot, David Cline, and Parris Egbert. "Importance Resampling for Global Illumination." In: *Eurographics Symposium on Rendering (2005)*. Ed. by Kavita Bala and Philip Dutre. The Eurographics Association, 2005. ISBN: 3-905673-23-1. DOI: 10.2312/EGWR/EGSR05/139-146.

[47] Philip Dutre, Philippe Bekaert, and Kavita Bala. *Advanced global illumination*. CRC Press, 2006.

[48] Philip Dutre, Philippe Bekaert, and Kavita Bala. *Advanced global illumination*. AK Peters/CRC Press, 2006.

[49] Michael Kass, Aaron Lefohn, and John Owens. *Interactive Depth of Field Using Simulated Diffusion on a GPU*. Tech. rep. Pixar, 2006.

[50] Earl Hammon Jr. "Practical Post-Process Depth of Field." In: *GPU Gems 3*. Ed. by Hubert Nguyen. Addison-Wesley, 2007. Chap. 28, pp. 583–606. ISBN: 9780321515261.

[51] Todd Kosloff and Brian Barsky. "An Algorithm for Rendering Generalized Depth of Field Effects Based on Simulated Heat Diffusion." In: *Proc. ICCSA*. 2007, pp. 1124–1140. ISBN: 978-3-540-74484-9. DOI: 10.1007/978-3-540-74484-9_99.

[52] M. Kraus and M. Strengert. "Depth-of-Field Rendering by Pyramidal Image Processing." In: *Proc. Eurographics*. 2007, pp. 645–654. DOI: 10.1111/j.1467-8659.2007.01088.x.

[53] Petrik Clarberg and Tomas Akenine-Möller. "Exploiting visibility correlation in direct illumination." In: *Computer Graphics Forum* 27.4 (2008), pp. 1125–1136. DOI: 10.1111/j.1467-8659.2008.01250.x.

[54] Paul Debevec. "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography." In: *ACM SIGGRAPH 2008 classes*. ACM. 2008, p. 32. DOI: 10.1145/1401132.1401175.

[55] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. "Multidimensional adaptive sampling and reconstruction for ray tracing." In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 33. DOI: 10.1145/1399504.1360632.

[56] Dylan Lacewell, Brent Burley, Solomon Boulos, and Peter Shirley. "Raytracing prefiltered occlusion for aggregate geometry." In: *2008 IEEE Symposium on Interactive Ray Tracing*. IEEE. 2008, pp. 19–26. DOI: 10.1109/RT.2008.4634616.

[57] Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi. "Real-Time Depth-of-Field Rendering Using Splatting on Per-Pixel Layers." In: *Computer Graphics Forum* 27.7 (2008), pp. 1955–1962. DOI: 10.1111/j.1467-8659.2008.01344.x.

[58] Steven Bergner, Mark S Drew, and Torsten Möller. "A tool to create illuminant and reflectance spectra for light-driven graphics and visualization." In: *ACM Transactions on Graphics (TOG)* 28.1 (2009), p. 5. DOI: 10.1145/1477926.1477931.

[59] Adrien Bousseau. *Non-Linear Aperture for Stylized Depth of Field*. SIGGRAPH 2009 - Technical talk. 2009. URL: http://artis.inrialpes.fr/Publications/2009/Bou09.

[60] Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Fredo Durand, and Ravi Ramamoorthi. "Frequency analysis and sheared reconstruction for rendering motion blur." In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 93. DOI: 10.1145/1540000.

[61] Shinya Kitaoka, Yoshifumi Kitamura, and Fumio Kishino. "Replica exchange light transport." In: *Computer Graphics Forum*. Vol. 28. 8. Wiley Online Library. 2009, pp. 2330–2342. DOI: 10.1111/j.1467-8659.2009.01540.x.

[62] Todd Kosloff, Michael Tao, and Brian Barsky. "Depth of Field Postprocessing For Layered Scenes Using Constant-Time Rectangle Spreading." In: *Proc. Graphics Interface*. 2009, pp. 39–46. URL: http://dl.acm.org/doi/10.5555/1555880.1555897.

[63] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. "Depth-of-field rendering with multiview synthesis." In: *ACM Transactions on Graphics (TOG)* 28.5 (2009), pp. 1–6. DOI: 10.1145/1618452.1618480.

[64] Sungkil Lee, Gerard J. Kim, and Seungmoon Choi. "Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation." In: *IEEE Trans. Vis. and Computer Graphics* 15.3 (2009), pp. 453–464. DOI: 10.1109/TVCG.2008.106.

[65] Cyril Soler, Kartic Subr, Fredo Durand, Nicolas Holzschuch, and Francois Sillion. "Fourier depth of field." In: *ACM Transactions on Graphics (TOG)* 28.2 (2009), p. 18. DOI: 10.1145/1516522.1516529.

[66] Wenzel Jakob. *Mitsuba renderer*. http://www.mitsuba-renderer.org. 2010.

[67] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. "Real-Time Lens Blur Effects and Focus Control." In: *ACM Trans. Graph.* 29.4 (July 2010). ISSN: 0730-0301. DOI: 10.1145/1778765.1778802.

[68] Kevin Egan, Florian Hecht, Fredo Durand, and Ravi Ramamoorthi. "Frequency analysis and sheared filtering for shadow light fields of complex occluders." In: *ACM Transactions on Graphics (TOG)* 30.2 (2011), p. 9. DOI: 10.1145/1944846.1944849.

[69] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. "Temporal light field reconstruction for rendering distribution effects." In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, p. 55. DOI: 10.1145/2010324.1964950.

[70] Anthony Pajot, Loïc Barthe, Mathias Paulin, and Pierre Poulin. "Combinatorial Bidirectional Path-Tracing for Efficient Hybrid CPU/GPU Rendering." In: *Computer Graphics Forum* 30.2 (2011), pp. 315–324. DOI: 10.1111/j.1467-8659.2011.01863.x.

[71] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. "A path space extension for robust light transport simulation." In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), p. 191. DOI: 10.1145/2366145.2366210.

[72] Wenzel Jakob and Steve Marschner. "Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport." In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 58. DOI: 10.1145/2185520.2185554.

[73] L McIntosh, Bernhard E Riecke, and Steve DiPaola. "Efficiently Simulating the Bokeh of Polygonal Apertures in a Post-Process Depth of Field Shader." In: *Computer Graphics Forum*. Vol. 31. 6. Wiley Online Library. 2012, pp. 1810–1822. DOI: 10.1111/j.1467-8659.2012.02097.x.

[74] Stefan Popov, Iliyan Georgiev, Philipp Slusallek, and Carsten Dachsbacher. "Adaptive quantization visibility caching." In: *Computer Graphics Forum* 32 (2013), pp. 399–408. DOI: 10.1111/cgf.12060.

[75] Justus Ulbrich, Jan Novák, Hauke Rehfeld, and Carsten Dachsbacher. "Progressive Visibility Caching for Fast Indirect Illumination." In: *Proceedings of the Vision, Modeling, and Visualization Workshop*. Ed. by Michael M. Bronstein, Jean Favre, and Kai Hormann. Lugano, Switzerland: Eurographics Association, 2013, pp. 203–210. URL: https://cg.ivd.kit.edu/english/PVCFID.php.

[76] Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. "Multiplexed metropolis light transport." In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 100. DOI: 10.1145/2601097.2601138.

[77] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. "On-line Learning of Parametric Mixture Models for Light Transport Simulation." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33.4 (2014). DOI: 10.1145/2601097.2601203.

[78] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. "On-line learning of parametric mixture models for light transport simulation." In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 101. DOI: 10.1145/2601097.2601203.

[79] Alexander Wilkie, Sehera Nawaz, Marc Droske, Andrea Weidlich, and Johannes Hanika. "Hero wavelength spectral sampling." In: *Computer Graphics Forum*. Vol. 33. 4. Wiley Online Library. 2014, pp. 123–131. DOI: 10.1111/cgf.12419.

[80] Yoshiharu Gotanda, Masaki Kawase, and Masanori Kakimoto. "Real-Time Rendering of Physically Based Optical Effects in Theory and Practice." In: *ACM SIGGRAPH 2015 Courses*. SIGGRAPH '15. Los Angeles, California: Association for Computing Machinery, 2015. ISBN: 9781450336345. DOI: 10.1145/2776880.2792715.

[81] Tim Mcgraw. "Fast Bokeh Effects Using Low-Rank Linear Filters." In: *Vis. Comput.* 31.5 (May 2015), pp. 601–611. ISSN: 0178-2789. DOI: 10.1007/s00371-014-0986-6.

[82] Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. "Probabilistic Connections for Bidirectional Path Tracing." In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 34.4 (2015). DOI: 10.1111/cgf.12680.

[83] Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. "Product Importance Sampling for Light Transport Path Guiding." In: *Computer Graphics Forum*. Vol. 35. 4. Wiley Online Library. 2016, pp. 67–77. DOI: 10.1111/cgf.12950.

[84] Hyuntae Joo, Soonhyeon Kwon, Sangmin Lee, Elmar Eisemann, and Sungkil Lee. "Efficient Ray Tracing Through Aspheric Lenses and Imperfect Bokeh Synthesis." In: *Computer Graphics Forum* 35.4 (2016), pp. 99–105. DOI: 10.5555/3071773.3071784.

[85] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)* 3rd. Morgan Kaufmann Publishers Inc., 2016.

[86] Philipp von Radziewsky, Thomas Kroes, Martin Eisemann, and Elmar Eisemann. "Efficient Stochastic Rendering of Static and Animated Volumes using Visibility Sweeps." In: *Transactions on Visualization and Computer Graphics* 23.9 (2016), pp. 2069 –2081. DOI: 10.1109/TVCG.2016.2606498.

[87] Jivri Vorba and Jaroslav Kvrivanek. "Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation." In: *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925912.

[88] M. Vorechovsk, V. Sadilek, and J. Elias. "Application of Voronoi Weights in Monte Carlo Integration with a Given Sampling Plan." In: *Proceedings of REC2016*. Inderscience Publishers. 2016, pp. 441–452.

[89] Ken Dahm and Alexander Keller. "Learning light transport the reinforced way." In: *ACM SIGGRAPH 2017 Talks*. ACM. 2017, p. 73. DOI: 10.1145/3084363.3085032.

[90] Kleber Garcia. "Circular Separable Convolution Depth of Field." In: *ACM SIGGRAPH 2017 Talks*. SIGGRAPH '17. Los Angeles, California: Association for Computing Machinery, 2017. ISBN: 9781450350082. DOI: 10.1145/3084363.3085022.

[91] Thomas Müller, Markus Gross, and Jan Novák. "Practical Path Guiding for Efficient Light-Transport Simulation." In: *Proceedings of the Eurographics Symposium on Rendering*. June 2017. DOI: 10.1111/cgf.13227.

[92] Jacopo Pantaleoni. "Charted metropolis light transport." In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 75. DOI: 10.1145/3072959.3073677.

[93] Wojciech Sterna. "Practical Gather-based Bokeh Depth of Field." In: *GPU Zen: Advanced Rendering Techniques*. Ed. by Wolfgang Engel. Bowker Identifier Services, 2017. Chap. 3, pp. 217–237. ISBN: 0998822892.

[94] Chakravarty R. Alla Chaitanya, Laurent Belcour, Toshiya Hachisuka, Simon Premoze, Jacopo Pantaleoni, and Derek Nowrouzezahrai. "Matrix Bidirectional Path Tracing." In: *Proceedings of the EGSR: EI&I*. Goslar, DEU: Eurographics Association, 2018, pp. 23–32. DOI: 10.2312/sre.20181169.

[95] Jerry Jinfeng Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. "Primary Sample Space Path Guiding." In: *Proceedings of the Eurographics Symposium on Rendering: EI&I*. SR '18. Eurographics Association, 2018, pp. 73–82. DOI: 10.2312/sre.20181174.

[96] Ricardo Marques, Christian Bouville, and Kadi Bouatouch. "Optimal Sample Weights for Hemispherical Integral Quadratures." In: *Computer Graphics Forum*. Wiley Online Library. 2018. DOI: 10.1111/cgf.13392.

[97] Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. "Geometry-Aware Metropolis Light Transport." In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275106.

[98] Florian Simon, Alisa Jung, Johannes Hanika, and Carsten Dachsbacher. "Selective guided sampling with complete light transport paths." In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37.6 (2018). DOI: 10.1145/3272127.3275030.

[99] Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. "Bayesian Online Regression for Adaptive Direct Illumination Sampling." In: 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201340.

[100] Tobias Zirr, Johannes Hanika, and Carsten Dachsbacher. "Re-Weighting Firefly Samples for Improved Finite-Sample Monte Carlo Estimates." In: *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, pp. 410–421. DOI: 10.1111/cgf.13335.

[101] Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P. A. Lensch, and Jaroslav Křivánek. "Volume Path Guiding Based on Zero-Variance Random Walk Theory." In: *ACM Trans. Graph.* 38.3 (2019), 25:1–25:19. DOI: 10.1145/3230635.

[102] Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. "Path Guiding in Production." In: *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. Los Angeles, California: ACM, 2019, 18:1–18:77. ISBN: 978-1-4503-6307-5. DOI: 10.1145/3305366.3328091.

[103] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (2020).

[104] Yuna Jeong, Seung Youp Baek, Yechan Seok, Gi Beom Lee, and Sungkil Lee. "Real-Time Dynamic Bokeh Rendering with Efficient Look-Up Table Sampling." In: *IEEE Trans. Visualization and Computer Graphics* Early Access (2021), pp. 1–12. DOI: 10.1109/TVCG.2020.3014474.

CURRICULUM VITAE

Jerry Jinfeng Guo 郭金锋



Jerry was born in Rizhao, China, where he holds his citizenship. He graduated from Rizhao Experimental High School (山东省日照实验高级中学) in 2008, where he trained for the 2008 College Entrance Exam (高考) and scored 640.

He then attended Dalian University of Technology (大连理工大学, DUT) major in *computer science and technology*. He interned at Crystal CG [1] between 2011 and 2012. In 2012, he graduated from DUT and joined Betop Multimedia Ltd. [2] as a technical director (pipeline TD, software engineer). In 2015, he continued his master study at Utrecht University (UU), major in *Game and Media Technology*. In 2017, he graduated from UU and started his PhD at Delft University of Technology. He got married in 2021 and became a father in 2022.

1 https://www.crystalcg.com/
2 https://betopjapan.co.jp/

LIST OF PUBLICATIONS

[1] Jerry Guo. *Towards More Robust and Efficient Monte Carlo Methods for Realistic Image Synthesis*. Closed conference event. Eurographics 2018 Doctoral Consortium, 2018.

[2] Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. "Primary Sample Space Path Guiding." In: *Eurographics Symposium on Rendering - EI & I*. Ed. by Wenzel Jakob and Toshiya Hachisuka. Eurographics. The Eurographics Association, 2018, pp. 73–82. DOI: 10.2312/sre.20181174.

[3] T. Callewaert, J. Guo, G. Harteveld, A. Vandivere, E. Eisemann, J. Dik, and J. Kalkman. "Multi-scale optical coherence tomography imaging and visualization of Vermeer's Girl with a Pearl Earring." In: *Optics Express* 28.18 (2020), pp. 26239–26256. DOI: 10.1364/OE.390703.

[4] Jerry Guo, Martin Eisemann, and Elmar Eisemann. "Next Event Estimation++: Visibility Mapping for Efficient Light Transport Simulation." In: *Computer Graphics Forum (Proc. of Pacific Graphics)* 39.7 (2020), pp. 205–217. DOI: 10.1111/cgf.14138.

[5] Jerry Guo and Elmar Eisemann. "Geometric Sample Reweighting for Monte Carlo Integration." In: *Computer Graphics Forum (Proc. of Pacific Graphics)* 40.7 (2021), pp. 109–119. DOI: 10.1111:cgf.14405.

[6] Jerry Guo, Markus Billeter, and Elmar Eisemann. "Effective CoC: Extended Aperture Model for Perspective Cameras in Rendering." In: *submission* (2022).

仰观宇宙之大，俯察品类之盛。 *Poetry by* Wang Xizhi *(303-361 AD), a poet from* Jin *dynasty, shows his humbleness and appreciation of the vastness of the universe and the abundance of beings.*

ACKNOWLEDGMENTS

This thesis is funded by NWO and research school ASCI.

According to my limited knowledge, there aren't many documents that can conclude one's four-five-year hard work. A PhD thesis is definitely one of them. I would like to start this final page with a brief look back along this fabulous journey, which, now that I think about it, goes way back.

My biggest influence in life is my family. My father, Guo Jianyong, provides me an impeccable childhood with plenty of opportunities to explore and build my curiosity which ultimately incubated my interest. My mom, Jin Liling, always shows me the beauty of paying attention to detail. I have to say that I could still use some effort in that regard. My older sister, Guo Wennan, sets a perfect model to look up to. I thank my family for raising me and showing me ways of life, and I hope that my humble contributions to the *stuff behind them video games and cartoons* (your original words) would make you proud.

Next up is my own small family. Meeting my wife, Sophia Sifan Wang, is the best thing that has ever happened to me. I thank you for the companion and the appreciation. Your optimistic nature adds the most colorful touch of my day. My new-born boy, Zilan, although some last tweeks of this thesis are done while I change your diapers, I couldn't wait to show you around. This thesis is dedicated to my family.

I also want to say a big thanks to the group of people that helped shaped my path during my early years in China. Especially, Li Fu, class 0809 guys back at DUT and guys from the students' union.

Coming to Utrecht, I've had the honor to have met and worked with many excellent people. I want to say thank you to Professor van Kreveld and Dr. Bikker at Utrecht University, for serious lectures I followed and casual discussions we had. Chongchuo, Yiran and Yunlu, thank you for the countless discussions and gatherings we had over the years, it's great to have you guys around. Fabian, Tom, Alex, thank you for keeping socially alive during the highly demanding course year back in 2015.

During my second year in Utrecht, I was searching for a rendering project for graduation thesis. Although I had a copy of Chinese version

of *Real-time Shadows*, I have never set foot on Delft soil. Anxiously, I send Elmar an email with a short description of what I have done and wanted to do, asking if he'd be available to supervise my master thesis. The next Monday, I got Elmar's reply in my mailbox, asking if I would be available to meet in person in two days. The meeting went surprisingly smooth as most of what I was saying was basically known to him. Everything went so fast that I only remember two things: coffee at EWI was bad and Elmar's super tall.

The next few years of working with Elmar has always been a charm. From the midnight discussions on the particular use of a word in a submission, to virtual meetings during the pandemic, besides the obvious input to the chapters in this PhD work, I really appreciate the open, candid, wise and deep interactions we had. I owe the most thank to Elmar, without whom this thesis wouldn't have been possible.

During the master project and first few months of my PhD, I had the honor to work with Dr. Pablo Bauszat, who sat down with me for countless times working on from equations to code. I really admire the professionalism. Pablo has shown me how to *defend* the honor as a *slow rendering guy*, which is such wonderful memory.

I also want to thank the colleagues who were with CGV before my arrival. Changgong and Jingtang are two Chinese colleagues who were about to defend their thesis when I arrived. Nevertheless, they were still great hosts and showed me around in the group. I wish you two the best with your careers back in China. Sungkil, I thank you for the tips you gave me during my early days in Delft, and also for sending over your implementation of the ray-traced DoF framework. Tim, I wish you and your family all the best in the States, I still owe you beer or a cat-sitting session for your proof-reading my submission draft (although given your review record, I doubt having a native speaker doing that is a good idea). Timothy, I thank you for the kindness you showed to new guys including me, it greatly helped me find that sense of belonging. Chaoran and Helwig, thank you for the good talks you gave during your stay at Delft. Victor, thank you for the interesting discussions we had over many offline rendering topics, I wish all the best to your career and that you are happy with the development. Big thank you to Julian, who has some of the best puzzles that only computer science guys could do. Nicola, thank you for receiving us at the Eindhoven light show 2018, it was freezing but really memorable. Niels, thank you for all the discussions we had over the years, you helped broaden my views and that's very important. Big thanks also to Christopher, Michael and Qiaomu, it has been a great honor working with you.

Special thanks for Markus, who is my co-promoter and coffee buddy. I admire your broad knowledge (although sometimes I am a bit worried

that you know some very specific details) and your calm attitude approaching things. I think I might have exploited you in the first few months you were in Leeds, where you started your current position. I hope the vacation in the alps could help at the time of writing.

Big thank you goes to Leo, who was basically always there since ever. I enjoy the conversations we had, although sometimes we don't always agree on some details, most turn out to be hilarious memory. Although I think it is very important that I take this opportunity to remind that you still owe us a house warming.

Equally big thank you goes to my peers, Nasikun, Annemieke, Baran and Xuejiao, who started about the same time as me. I really appreciate the good interactions we had over the years and really admire you for your individual traits. I wish all the best to Nasikun, who has defended before me and will continue his career in Indonesia. Big applause to Baran, Annemieke and Xuejiao, you have already conducted excellent work and I sincerely look forward to your output.

To the bachelor and master students that I have worked closely with: Guusje, Mauro, Tan and Peter Guowei, it has been a great honor having worked with you and I have really enjoyed it. To see your work coming to shape is beyond delight. I sincerely hope the best for your next steps in career or study.

Special thanks for the faculty members of CGV, Rafa, Anna, Klaus, Thomas and Ricardo. Thank you each for all the tips and feedback you gave over the years. Big thanks for Marloes and Lauretta for helping me settle so many administration work. Fat thanks for Ruud and Bart, who helped me over countless technical issues.

I would also like to thank the relatively new colleagues, whose arrival have been overshadowed by either the pandemic or relocation as most of the time we only meet virtually. Michael, Petr, Mark(Dr.) and Martin, I wish you all the best with your career at CGV. Peiteng, Alex, Marcos, Mijael, Faizan, Mark, Ruben, Mathijs, Yang, Nicolas, Ali, Soumyadeep, Chang and Peter Guowei, I wish you all the best with your PhD journey at CGV.

This is officially the final sentence of the thesis, to conclude which I wanted to say thank *you*, dear reader: I hope you find something useful from the work that took good four-five years of my life.