Optimized Shift Mapping Operators for Rendering

by

Terry Yu-Xi CHEN

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN INFORMATION TECHNOLOGY ENGINEERING
M.A.Sc.

MONTREAL, AUGUST 10, 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

© Copyright 2026 reserved by Terry Chen

© Copyright reserved
It is forbidden to reproduce, save or share the content of this document either in whole or in parts. The reader who wishes to print or save this document on any media must first get the permission of the author.

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Adrien Gruson, Thesis Supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Derek Nowrouzezahrai, Thesis Co-Supervisor
Electrical and Computer Engineering, McGill University

Mr. Sheldon Andrews, Chair, Board of Examiners
Department of Software and IT Engineering, École de technologie supérieure

Mr. Guillaume Gilet, External Examiner
Department of Computer Science, Université de Sherbrooke

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON AUGUST 3, 2026

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisors, Prof. Adrien Gruson and Prof. Derek Nowrouzezahrai, for giving me the opportunity to work with them, and for their support and direction during my graduate studies. I would also like to thank my labmates from the Multimedia lab for their moral support and the many insightful research discussions we had together.

I am grateful to the team at Ubisoft La Forge for accepting me as an intern, and for introducing me to the wonders of real-time rendering research. In particular, I would like to thank the environment team of Arnaud Schoentgen, Damien Rioux-Lavoie, Hisanari Otsu, and Jean-Phillipe Guertin. This thesis would not have been possible without their mentorship and guidance.

My friends from MCAGC, McGill GCF, BTBC, and elsewhere (too many to mention!) made the best company that I could've asked for during my time in graduate school. I am greatly thankful for each and every one of them for their prayers and spiritual support, their companionship through my personal struggles, and for making life a little more bearable during the hardships of research and writing.

Ernest Lau, Jeffrey Liu, Joshua Issa, and Justin Benoit helped review this thesis. I would like to thank them for their insightful comments, and for catching all the embarrassing errors I made.

Finally, I would like to thank my parents for always being there for me through the ups and downs of life. This thesis would not have been possible without their love, patience, and sacrifice, and I am deeply indebted to them for their support.

Opérateurs de décalage optimisés pour le rendu

Terry Yu-Xi CHEN

RÉSUMÉ

Le rendu réaliste basé sur la physique repose sur une simulation efficace du transport de la lumière, pour laquelle les algorithmes coûteux de lancer de chemins constituent la seule solution efficace. Ce coût résulte du grand nombre d'échantillons de chemins nécessaires pour générer une image exploitable, ce qui rend cette approche irréalisable pour les applications en temps réel. ReSTIR effectue une réutilisation spatio-temporelle des chemins, réduisant considérablement le nombre d'échantillons requis à un niveau compatible avec le matériel graphique moderne. Son efficacité dépend de la qualité de ses opérateurs de décalage, qui transforment les chemins entre les domaines de chemins distincts de différents pixels et images. Cependant, les opérateurs de décalage actuels ne tiennent pas compte de la qualité de leurs échantillons sources, ce qui entraîne la réutilisation d'échantillons non contributifs et une augmentation du bruit. Dans cette thèse, je présente un opérateur de décalage optimisé basé sur le mécanisme du filtrage stochastique de texture, qui améliore considérablement la qualité des échantillons de chemins disponibles pour la réutilisation spatio-temporelle des chemins.

Mots-clés: rendu en temps réel, rééchantillonnage préférentiel, ReSTIR, filtrage de texture

Optimized Shift Mapping Operators for Rendering

Terry Yu-Xi CHEN

ABSTRACT

Realistic physically-based rendering relies on efficient light transport simulation, for which expensive path tracing algorithms are the only effective solution. This cost arises from the large number of path samples required to generate a usable image, which makes it infeasible for real-time applications. ReSTIR performs spatiotemporal path reuse, drastically reducing the required number of samples to within the realm of modern graphical hardware. Its efficiency is conditional on the quality of its shift mappings, which transform paths between the distinct path domains of different pixels and frames. However, current shift mappings do not consider the quality of their source samples, resulting in reuse of non-contributing samples and increased noise. In this thesis, I present an optimized shift mapping operator based on the machinery of stochastic texture filtering, which significantly increases the quality of path samples available for spatiotemporal path reuse.

Keywords: real-time rendering, resampled importance sampling, ReSTIR, texture filtering

TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 BACKGROUND 5
1.1 Light transport simulation 5
1.1.1 Common BSDFs 7
1.2 Monte Carlo integration 8
1.3 Path tracing 10
1.3.1 Local sampling for practical path construction 13
1.4 Variance reduction 15
1.4.1 Importance sampling 16
1.4.2 Multiple importance sampling 18
1.4.3 Resampled importance sampling 20
1.5 Shift mappings 22
1.6 Real-time global illumination 24

CHAPTER 2 RESERVOIR-BASED SPATIOTEMPORAL IMPORTANCE RESAMPLING 27
2.1 Introduction 27
2.2 Method 28
2.2.1 Weighted reservoir sampling 28
2.2.2 Reservoir resampling 30
2.3 ReSTIR in practice 32
2.3.1 Confidence weighting 35
2.3.2 Advanced MIS weights 35
2.3.3 Advanced shift mappings 38
2.4 ReSTIR variations 40
2.4.1 ReSTIR DI 40
2.4.2 ReSTIR GI 41
2.4.3 ReSTIR PT 41
2.4.4 Other improvements 42
2.5 Discussion 43

CHAPTER 3 MIPMAPPED RESERVOIRS 45
3.1 Introduction 45
3.2 Background 46
3.2.1 Texture filtering 46
3.2.2 Stochastic texture filtering 47
3.2.3 Mipmapping 48
3.3 Reservoir mipmapping 50
3.3.1 Reservoir merging 53

3.3.2 Reservoir selection 56
3.3.3 Adaptive clustering 57
3.4 Results 60

CONCLUSION AND RECOMMENDATIONS 69

APPENDIX A RESERVOIR REGULARIZATION WITH VIRTUAL SPHERICAL GAUSSIAN LIGHTS 71

APPENDIX B BOILING ARTIFACTS IN SPATIAL REUSE 79

APPENDIX C PROOFS 81

APPENDIX D CODE LISTINGS 83

LIST OF REFERENCES 87

LIST OF FIGURES

Page

Figure 1.1 A light path $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$ with sensor vertex $\mathbf{x}_0$ and terminal light vertex $\mathbf{x}_4$. The layout of the scene means that $\mathbf{x}'_3$ cannot be successfully sampled from $\mathbf{x}_2$ as it is occluded by $\mathbf{x}_3$ 6

Figure 1.2 Materials and their BSDFs 7

Figure 1.3 Despite being in separate disconnected rooms (orange and blue), **y** and **z** can both be sampled from **x** in an area-domain sampling strategy. However, with solid angle sampling it is impossible to select **y** from **x**, thus preserving physical limitations 13

Figure 1.4 **(a)** A uniform distribution can differ significantly from the integrand, which increases variance. **(b)** To counteract this, importance sampling uses a distribution that closely matches the integrand, allowing f/p to be more consistent across samples 16

Figure 1.5 Comparing the results of **(a)** BSDF sampling, **(b)** NEE sampling, and **(c)** MIS between the two strategies Taken from Veach (1997, chapter 9) . 17

Figure 1.6 **(a)** MIS weighs the contributions of the candidate samples to output an efficient estimator. **(b)** On the contrary, RIS selects one of the candidate samples, eventually producing a sample pool with distribution equal to the target function 20

Figure 1.7 **(a)** The path space for pixel j_1 varies from that for pixel j_2 as their primary hit points differ in position, which in turn influences the domain of the sampled secondary ray. **(b)** Shift mappings allow us to evaluate paths from j_1 in the domain of j_2, allowing for reuse 22

Figure 1.8 Reconnection shift mapping 23

Figure 2.1 Visualization of updating the reservoir in WRS. Here, the reservoir Y has chosen X_i, and we are testing if X_j should replace it. Replacement only succeeds if $\text{Unif}[0, 1] < \frac{w_j}{\sum w_k}$, otherwise X_i remains in the reservoir 28

Figure 2.2 Equal-time (5 s) comparison between path tracing, spatial reuse, and spatial+temporal reuse on the Zero Day scene 34

Figure 2.3 Various shift mappings used in ReSTIR. We showcase **(a)** reconnection shift, **(b)** random sequence replay, **(c)** half-vector copy, and **(d)** the hybrid random sequence replay/reconnection mapping introduced in Hua *et al.* (2019) 39

Figure 3.1 Example of a mipmap, with the original image at left. Note the lack of aliasing artifacts in each progressively downsampled level 49

Figure 3.2 **(a)** ReSTIR has no control over the spatial samples that it selects, which can lead to attempts to reuse low-quality samples. **(b)** By grouping reservoirs into 2×2 clusters à la mipmapping, the likelihood of choosing a low-quality sample is lessened. The target density and weights of each cluster can be calculated through **(c)** reservoir selection, where one of the four reservoirs is chosen to represent the entire cluster, or **(d)** reservoir merging, which produces a single representative reservoir 50

Figure 3.3 **(a)** Reservoir selection and **(b)** reservoir merging performed over multiple mipmap levels 52

Figure 3.4 **(a)** Fixing the location of the canonical sample results in highly visible correlation artifacts propagating down the mip chain, worsening the quality of the final image. **(b)** Uniformly choosing the canonical samples reduces the maximum intensity of these artifacts. The bottom rows depict the SMAPE metric in false colour. Scene is a closed Cornell Box supplemented with the Stanford Dragon 55

Figure 3.5 **(a)** In a screen-space mipmap grid, a texel can contain geometric discontinuities when it covers multiple objects or primitives. **(b)** Adaptive clustering adjusts for the scene geometry, and can also be made aware of surface orientation 57

Figure 3.6 One-sample renders over various scenes, comparing ReSTIR GI with level-3 equivalents of reservoir merging, reservoir selection, and adaptive clustering 62

Figure 3.7 Comparison of different mip levels on the Zero Day scene. As mip level increases, variance decreases, but bubbling is more strongly visible 63

Figure 3.8 Breakdown of results for specular surfaces and caustics. Path length 3, indirect lighting, 1 sample. Lowest ꟻLIP score for each crop indicated with inverted colours. Note that selection performs the best on specular surfaces, while mipmapping performs worse especially as the mipmap level increases; however, when rendering caustics, the latter method delivers the best results 64

Figure 3.9 Equal-time (10 s) rendering of the Staircase scene comparing level-3 reservoir selection versus SLIC8, with weights $w_g = 1$ and $w_{ss} = 0.001$ for the latter. All error values shown in SMAPE. False-colour values scaled 5× for contrast 65

Figure 3.10 Equal-time convergence plots for spatial reuse over our test scenes, measuring SMAPE versus render time. We compare against ReSTIR GI without temporal reuse as our baseline 66

Figure 3.11 Bias shown by Sel3 on Living Room and Bistro respectively after accumulating for 10 seconds. Red regions denote negative bias, green regions positive bias, and yellow regions mixed bias between RGB channels 67

LIST OF ALGORITHMS

Page

Algorithm 1.1 Path tracing 15

Algorithm 2.1 Reservoir RIS 29

Algorithm 2.2 Reservoir merging 30

Algorithm 2.3 Generalized MIS and RIS weights 31

Algorithm 2.4 ReSTIR in practice 33

Algorithm 2.5 ReSTIR with confidence weights 36

Algorithm 3.1 Reservoir mipmapping 51

Algorithm 3.2 Mipmapped reservoir merging 54

Algorithm 3.3 Strategies for choosing the canonical reservoir 54

Algorithm 3.4 Mipmapped reservoir selection 56

Algorithm 3.5 SLIC-based reservoir selection 58

LIST OF ABBREVIATIONS

BSDF	Bidirectional scattering distribution function
GRIS	Generalized resampled importance sampling
MC	Monte Carlo
MIS	Multiple importance sampling
NEE	Next event estimation
RIS	Resampled importance sampling
ReSTIR	Reservoir-based spatiotemporal importance resampling
SG	Spherical Gaussian
SLIC	Simple linear iterative clustering
SMAPE	Symmetric mean absolute percentage error
STF	Stochastic texture filtering
UCW	Unbiased contribution weight
VPL	Virtual point light
VSL	Virtual sphere light
VSGL	Virtual spherical Gaussian light
WRS	Weighted reservoir sampling

INTRODUCTION

Rendering is the art of generating an image from a scene description. In order to produce images that accurately represent real-life phenomena, physically accurate mathematical models are used to solve the rendering problem, leading to the field of physically-based rendering. However, these methods are too expensive for real-time applications where minimizing latency and computation time is prioritized, thus motivating cheaper approximations for photorealistic shading and global illumination. Despite this, physically-based rendering remains the gold standard of rendering approaches, and modern hardware advances have improved its feasibility in real-time applications.

One physically-based rendering technique is light transport simulation, which tracks the physical scattering of light within a given scene with a recursive integral equation (Kajiya, 1986). In practice, this integral equation is nonelementary and hence cannot be evaluated analytically, forcing the use of numerical methods like Monte Carlo integration, which is notable for being asymptotically correct and embarrassingly parallel. Applied together, these two techniques form the basis of *path tracing*, the preferred solver for the rendering equation owing to its efficiency and robustness across a variety of lighting scenarios (Veach, 1997).

Unfortunately, as a stochastic algorithm Monte Carlo integration produces unwanted variance artifacts that are visible as noise. Although denoising algorithms can remove most of the noise (Munkberg & Hasselgren, 2020), their performance suffers if the inputs are too noisy. The simplest way to reduce the amount of noise is to increase the number of samples used in Monte Carlo integration, but this incurs a large computational cost incompatible with real-time applications.

The most impactful optimizations to the path tracing algorithm thus take the form of variance reduction methods, which seek to reduce the amount of variance per sample and hence the total number of samples required to generate a high-quality render. While previous approaches to variance reduction have focused on improving the efficiency of sample generation (Veach & Guibas, 1995; Jensen, 1996; Müller, Gross & Novák, 2017), a new avenue has recently

opened up in the field of real-time sample reuse. Reservoir-based spatiotemporal importance resampling (ReSTIR), introduced in Bitterli *et al.* (2020), combines temporally and/or spatially coherent samples to generate a single sample that has the statistical power of exponentially more samples, substantially reducing the amount of per-sample variance.

Despite these improvements, ReSTIR is still highly dependent on the quality of its source samples. Furthermore, it depends on shift mappings to shift samples between pixel domains which increase variance when they fail, which is often. Our goal is hence to decrease the failure rate of shift mappings, in turn increasing the robustness of the sample reuse process. We achieve this with our mipmapped reservoir framework, which successfully increases the quality of samples available to ReSTIR without significantly modifying the core algorithm, thus providing an easily implemented method to further reduce variance.

0.1 Summary of contributions

We present a novel mipmapped reservoir framework that significantly enhances the effectiveness of spatial reuse in ReSTIR, based on the principles of stochastic texture filtering (Pharr, Wronski, Salvi & Fajardo, 2024) and mipmapping (Williams, 1983). We build upon the former to filter for high-quality samples within the reservoir buffer, and use the latter to hierarchically construct a chain of downsampled reservoir buffers. We derive reservoir merging as an unbiased technique within this framework, and discuss its cost and implementation considerations. We also derive reservoir selection as a cheaper approximation, alongside a novel adaptive cluster importance sampling mechanism which adapts to the geometric and material properties of the scene to enhance path similarity when performing sample reuse.

As a complementary method to the ReSTIR algorithm, implementation of mipmapped reservoirs in a rendering pipeline only requires adding a few extra passes to handle the mipmapping process. The core data flow of sample generation and reservoir reuse remains untouched without needing to substantially modify the neighbour selection process. We demonstrate this ease of application, and that mipmapped reservoirs significantly reduces ReSTIR's output variance at little extra cost.

In addition to mipmapped reservoirs, we also investigate reservoir regularization with virtual spherical Gaussian lights (Tokuyoshi, 2015). We show that it is possible to build spherical Gaussians on-the-fly from the information stored in the reservoirs to regularize the shift mappings used in ReSTIR, in turn reducing the intensity of fireflies. We present our method and its effectiveness in a separate Appendix A.

0.2 Structure of thesis

In Chapter 1, we cover the basic theory of light transport simulation, Monte Carlo integration, and the path tracing algorithm. We continue by discussing variance reduction methods, and introduce both multiple importance sampling (Veach & Guibas, 1995) and resampled importance sampling (Talbot, Cline & Egbert, 2005). We further introduce the theory of shift mappings, and conclude with an overview of real-time techniques for solving the global illumination problem.

In Chapter 2, we introduce the basic ReSTIR algorithm as outlined in Wyman *et al.* (2023). We explain how weighted reservoir sampling and resampled importance sampling can be combined into a constant-memory online sampling algorithm. We apply this combination to the rendering loop, demonstrate its variance reduction capabilities, and discuss practical considerations for implementing ReSTIR. Finally, we discuss state-of-the-art improvements to ReSTIR and its current weaknesses.

Chapter 3 presents the theory and design of our mipmapped reservoir framework. We begin by covering the theory of texture filtering and mipmapping, and how they can be applied to ReSTIR. We then derive the reservoir merging and reservoir selection techniques. We introduce our adaptive clustering mechanism and show its utility within our framework. We conclude by demonstrating the significant improvements in sample quality and variance reduction that our methods provide across a variety of lighting scenarios at very little added cost.

Part of the research presented in this thesis was conducted at Ubisoft La Forge in Montreal, Quebec, Canada. Ubisoft claims no ownership over the corresponding text, implementation, or any derivative works thereof.

CHAPTER 1

BACKGROUND

1.1 Light transport simulation

It is an intuitive fact of nature that objects are only made visible if they reflect light into an optical sensor, be it the viewer's eye or a camera. Modelling how light reflects and bounces around a scene, however, is much less intuitive. Light transport simulation mimics the physical phenomena of light interacting with a three-dimensional scene by tracing the directions and surface intersections of photons from the source up until they reach the sensor. The direction that a photon bounces to at a given surface is governed by the *bidirectional scattering distribution function* (BSDF), an angular statistical distribution that also defines the surface's appearance, or its material (Pharr, Jakob & Humphreys, 2017). The bidirectional nature of BSDFs ensures that light transport simulation can be conducted from the sensor as well, with photons being traced until an emissive source is reached. To start, let $\mathbf{x}$ be a point visible to the sensor located at $\mathbf{p}$ in the scene, where visibility is determined by shooting a ray originating from the sensor and retrieving the closest surface point along the ray, and denote directions as being in differential solid angle over the sphere S^2. Then the outgoing contribution of $\mathbf{x}$ in the sensor direction $\omega_o = (\mathbf{p} - \mathbf{x})/\|\mathbf{p} - \mathbf{x}\|$ takes the form of the rendering equation from Kajiya (1986):

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{S^2} f_r(\mathbf{x}, \omega_o, \omega_i) L(\mathbf{y}, -\omega_i) |\mathbf{n}_\mathbf{x} \cdot \omega_i| \, \mathrm{d}\omega_i \,, \tag{1.1}$$

where L is the incoming radiance, L_e is the emitted radiance at $\mathbf{x}$ and is nonzero if $\mathbf{x}$ is on a light source, f_r is the BSDF, ω_i is the incoming light ray direction at $\mathbf{x}$, $\mathbf{y}$ is the scene intersection with the ray with origin $\mathbf{x}$ and direction ω_i, and $\mathbf{n}_\mathbf{x}$ is the normal at $\mathbf{x}$. Alternatively, we can choose to integrate over all points $\mathbf{y}$ in the scene and rewrite the above as

$$L(\mathbf{p} \leftarrow \mathbf{x}) = L_e(\mathbf{p} \leftarrow \mathbf{x}) + \int_A f_r(\mathbf{y} \rightarrow \mathbf{x} \rightarrow \mathbf{p}) G(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{x} \leftarrow \mathbf{y}) \, \mathrm{d}A(\mathbf{y}) \,. \tag{1.2}$$

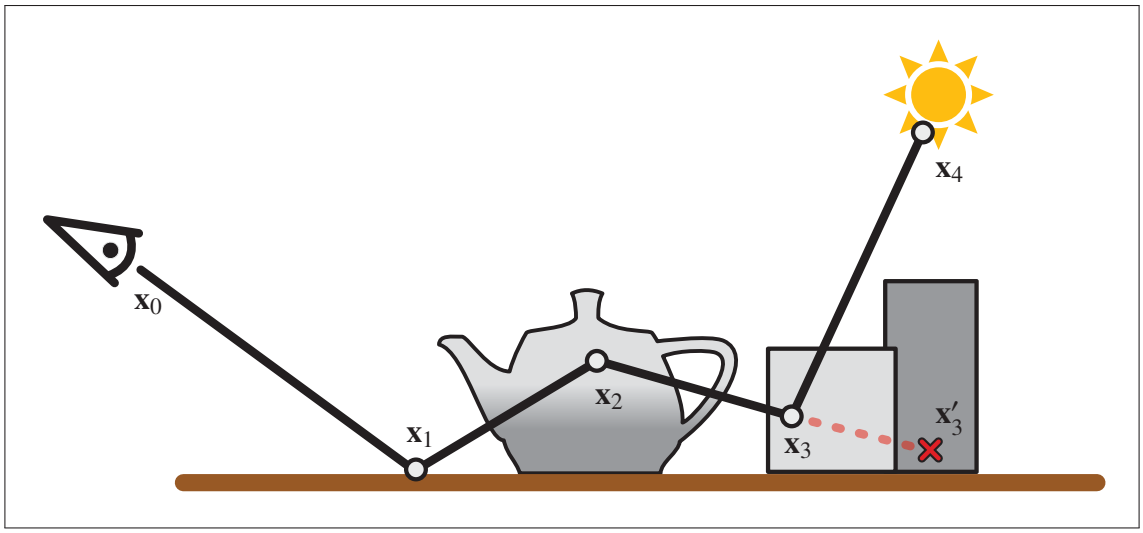


Figure 1.1 A light path $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$ with sensor vertex $\mathbf{x}_0$ and terminal light vertex $\mathbf{x}_4$. The layout of the scene means that $\mathbf{x}'_3$ cannot be successfully sampled from $\mathbf{x}_2$ as it is occluded by $\mathbf{x}_3$

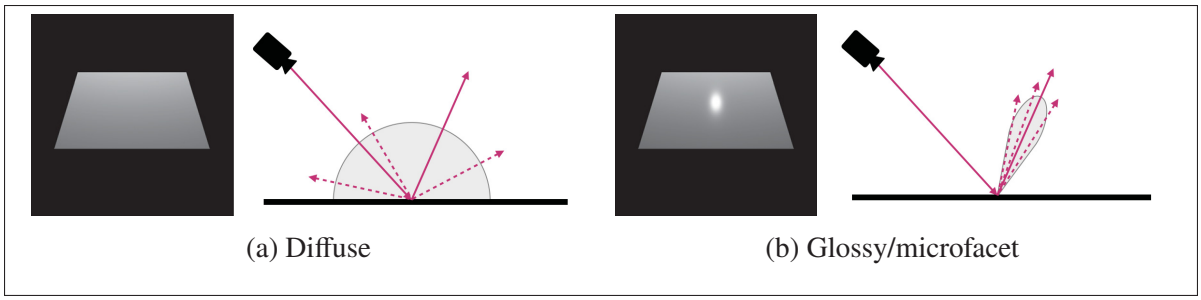
Changing the sampling domain to area measure introduces a Jacobian that when combined with the cosine term $|\mathbf{n}_\mathbf{x} \cdot \omega_i|$ gives the geometry term $G(\mathbf{x} \leftrightarrow \mathbf{y})$:

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = V(\mathbf{x} \leftrightarrow \mathbf{y}) \frac{\left|\mathbf{n}_\mathbf{x} \cdot \omega_{\mathbf{x}\to\mathbf{y}}\right| \left|\mathbf{n}_\mathbf{y} \cdot \omega_{\mathbf{y}\to\mathbf{x}}\right|}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{1.3}$$

where $V(\mathbf{x} \leftrightarrow \mathbf{y})$ is 1 if $\mathbf{x}$ and $\mathbf{y}$ are visible to each other and 0 otherwise, and $\omega_{\mathbf{x}\to\mathbf{y}} = (\mathbf{y} - \mathbf{x})/\|\mathbf{y} - \mathbf{x}\|$ is the unit direction from $\mathbf{x}$ to $\mathbf{y}$. If we generalize $\mathbf{p}$, $\mathbf{x}$, and $\mathbf{y}$ to arbitrary points, we get the area-domain formulation of the rendering equation:

$$L(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i) = L_e(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i) + \int_A f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) L(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) \, \mathrm{d}A(\mathbf{x}_{i+1}) \, . \tag{1.4}$$

This allows us to construct a *path* $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k]$, where $\mathbf{x}_0 = \mathbf{p}$ represents the sensor. We see an example of visibility coming into play in Figure 1.1: if we are currently at $\mathbf{x}_2$, we can sample $\mathbf{x}_3$ but not $\mathbf{x}'_3$ as the latter is occluded.



(a) Diffuse (b) Glossy/microfacet

Figure 1.2 Materials and their BSDFs

A closed-form solution for the rendering equation is mainly predicated on whether analytic integration of the BSDF f_r is possible. It thus behooves us to discuss common choices for the BSDF and their properties.

1.1.1 Common BSDFs

On top of being statistical distributions, BSDFs are also phenomenological quantifiers for modelling the look and colour of the physical material at a given surface point. This means that the choice of BSDFs significantly impacts the shading of the scene, and hence the final generated image. BSDFs are usually split into diffuse and specular components. The diffuse component is the simplest, and models the case where the surface scatters incoming light perfectly evenly; that is, outgoing rays are uniformly distributed on the hemisphere, which we visualize in Figure 1.2a. Materials with BSDFs that are comprised entirely of the diffuse component, such as a wall covered with matte paint, have the advantage of being easy to sample from and evaluate, and because their lobe (i.e. distribution) covers the entire sampling domain, they can provide particularly efficient estimators. On the other hand, the specular component is associated with a tighter distribution, since reflected rays are concentrated within a smaller solid angle to allow for specular highlights, as we see in Figure 1.2b. The size of the lobe corresponds with the BSDF's *specularity*, with a higher specularity corresponding to a smaller lobe; these BSDFs correspond to glossy materials such as smooth plastic or metal surfaces. At the extreme end, a specular BSDF follows a Dirac delta distribution with mass located at the reflection vector, essentially acting as a perfect mirror.

There exist a large number of families of BSDFs that are able to accurately represent real-world materials (Pharr *et al.*, 2017). Microfacet BSDFs (Walter, Marschner, Li & Torrance, 2007), which are derived from micro-scale light transport simulation, serve as one of the more popular families, adding a normal distribution function (NDF) and a shadowing-masking function to the standard BSDF formulation. The former function simulates how microscopic surface details alter the reflected direction of light, while the latter models the self-shadowing and occlusion that occurs from those same details. A roughness parameter is then used to control the width of the NDF, inversely correlating to the BSDF's specularity. The accuracy and control thus offered make microfacet BSDFs the preferred model for physically-based rendering applications (Burley & Walt Disney Animation Studios, 2012). Unfortunately, they are also difficult to sample from (Heitz, 2018; Dupuy & Benyoub, 2023).

Classical microfacet models do not have an analytical formulation that permits evaluation of the light contribution of the scattering (Wang *et al.*, 2022; Heitz, Hanika, d'Eon & Dachsbacher, 2016). The complicated nature of microfacet BSDFs and the recursive nature of the rendering equation, as visible in Equation (1.1), means that analytical evaluation of the latter is impossible in practice, since knowing the outgoing value of $L(\mathbf{x}, \omega_o)$ requires knowing the incoming value $L(\mathbf{y}, -\omega_i)$. As a result, numerical integration methods must be used to evaluate the rendering equation.

1.2 Monte Carlo integration

Evaluating an integral with no analytical antiderivative is a technically challenging proposition. Conventional numerical integration methods such as Newton-Cotes and finite difference methods work well in the majority of situations, and are commonly used in scientific and engineering applications (Davis & Rabinowitz, 2007). However, these methods fail in the high-dimensional rendering context as they rely on discretisation of the problem domain, which introduces unwanted bias and invokes the curse of dimensionality, leading to significant memory costs. Monte Carlo integration sidesteps these issues by introducing an asymptotically unbiased

stochastic estimator, and is the preferred method for solving the rendering equation owing to its embarrassingly parallel nature (Kontoghiorghes, 2005).

Given a continuous random variable X with support Ω, its expected value is defined as

$$\mathbb{E}[f(X)] = \int_{\Omega} f(x)p(x)\mathrm{d}x. \tag{1.5}$$

Hence, given an integral

$$I = \int f(x)\mathrm{d}x, \tag{1.6}$$

we have that $\mathbb{E}[f(X)/p(X)] = I$ as long as $p(X) \neq 0$ wherever $f(X) \neq 0$. Given realized samples $x_1, \cdots, x_n \overset{iid}{\sim} p(X)$, we thus derive the practical Monte Carlo estimator

$$\langle I \rangle_{MC} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}. \tag{1.7}$$

As an estimator, it is possible for $\langle I \rangle_{MC}$ to deviate from the true value of I, which we model as bias. The MC estimator is strictly unbiased and therefore accurate, meaning that the estimated value of $\langle I \rangle_{MC}$ is equal to the desired integral:

$$\mathbb{E}[\langle I \rangle_{MC}] = \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}\right] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}\left[\frac{f(X_i)}{p(X_i)}\right] = \frac{1}{N} \sum_{i=1}^{N} I = I. \tag{1.8}$$

Because $\langle I \rangle_{MC}$ is a stochastic estimator, evaluating it will introduce error which we seek to minimize. This error is modelled as variance, defined for a random variable X as

$$\mathrm{Var}(X) = \mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2. \tag{1.9}$$

By minimizing variance, we can derive more precise results with fewer samples. The variance of $\langle I \rangle_{MC}$ is

$$\mathrm{Var}(\langle I \rangle_{MC}) = \mathrm{Var}\left(\frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}\right) = \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}\left(\frac{f(X_i)}{p(X_i)}\right) = \frac{1}{N} \mathrm{Var}\left(\frac{f(X)}{p(X)}\right), \tag{1.10}$$

where the final simplification is possible due to the X_i being independent and identically distributed. Since variance is error squared, this means that Monte Carlo integration converges at a rate of $O(1/\sqrt{N})$; in other words, an n-fold reduction in error requires $O(n^2)$ additional samples. The above also implies that error is minimized when $\mathrm{Var}(f(X_i)/p(X_i))$ is close to 0, i.e. when the individual values of $f(X_i)/p(X_i)$ are close to each other. Specifically, samples where $f(X_i) = 0$ increase variance if $I \neq 0$, which makes them wasted as they contribute nothing to the final estimate, an observation we will apply throughout this chapter.

1.3 Path tracing

By combining light transport simulation, the notion of a light path, and Monte Carlo integration, we derive the *path tracing* algorithm for solving the rendering equation (Pharr *et al.*, 2017). Recall the area-domain formulation from Equation (1.4):

$$L(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i) = L_e(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i) + \int_A f_r(\mathbf{x}_{i+1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i-1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) L(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) \,\mathrm{d}A(\mathbf{x}_{i+1}) \,.$$

Although we could attempt to apply Monte Carlo integration to this specific integral equation, we would soon find ourselves confounded by its recursive nature. Thus, we expand the recursion starting from $i = 1$:

$$\begin{aligned} L(\mathbf{x}_0 \leftarrow \mathbf{x}_1) = {} & L_e(\mathbf{x}_0 \leftarrow \mathbf{x}_1) \\ & + \int_A f_r(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) L_e(\mathbf{x}_1 \leftarrow \mathbf{x}_2) \,\mathrm{d}A(\mathbf{x}_2) \\ & + \iint_A f_r(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\ & \qquad \times f_r(\mathbf{x}_3 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_1) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3) L_e(\mathbf{x}_2 \leftarrow \mathbf{x}_3) \,\mathrm{d}A(\mathbf{x}_3) \,\mathrm{d}A(\mathbf{x}_2) \\ & + \cdots \,. \end{aligned} \tag{1.11}$$

Clearly, this expansion results in an infinite sum $\sum_{i=1}^{\infty} P(\bar{\mathbf{x}}_k)$, where $\bar{\mathbf{x}}_k = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k]$ is an arbitrary path of length k and

$$P(\bar{\mathbf{x}}_k) = \underbrace{\int_A \cdots \int_A}_{k-1 \text{ times}} \left(\prod_{i=1}^{k-1} f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i) \right) \times L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k) \, \mathrm{d}A(\mathbf{x}_k) \cdots \mathrm{d}A(\mathbf{x}_2) \, . \tag{1.12}$$

The parenthesized expression gives us the key definition of the *throughput* function, which is the incrementally constructed portion of the integrand:

$$T(\bar{\mathbf{x}}_k) = \prod_{i=1}^{k-1} f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i). \tag{1.13}$$

By defining $\mathcal{P}_k$ as the set of all paths of length k and performing integration over $\bar{\mathbf{x}}_k$, we rewrite the given infinite sum as

$$L(\mathbf{x}_0 \leftarrow \mathbf{x}_1) = L_e(\mathbf{x}_0 \leftarrow \mathbf{x}_1) + \sum_{k=2}^{\infty} \int_{\mathcal{P}_k} T(\bar{\mathbf{x}}_k) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k) \, \mathrm{d}\bar{\mathbf{x}}_k \, . \tag{1.14}$$

In practice, each light path $\bar{\mathbf{x}}_k$ must be associated with a pixel, and we tend to perform per-pixel MC estimation. This requires us to evaluate all contributing light paths at each pixel, which in turn requires us to formally define the relation between a given pixel and a primary hit $\mathbf{x}_1$. We thus introduce the pixel importance function $W_e^j(\mathbf{x}_0, \mathbf{x}_1)$, which is nonnegative if $\mathbf{x}_1$ corresponds to pixel j of the sensor centred at $\mathbf{x}_0$ and 0 otherwise, hence filtering for valid camera subpaths $\mathbf{x}_0 \to \mathbf{x}_1$. Letting $\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}_k$ and rewriting the above to be dependent solely on j, we thus derive the *path-space formulation* of the rendering equation (Veach, 1997):

$$I_j = \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T(\bar{\mathbf{x}}_k) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k) \, \mathrm{d}\bar{\mathbf{x}}_k \, . \tag{1.15}$$

This provides a convenient starting point for Monte Carlo integration, which gives us the estimator

$$\langle I_j \rangle_{MC} = \frac{1}{N} \sum_{i=1}^{N} \frac{W_e^j(\mathbf{x}_{i,0}, \mathbf{x}_{i,1}) G(\mathbf{x}_{i,0} \leftrightarrow \mathbf{x}_{i,1}) T(\bar{\mathbf{x}}_{i,k}) L_e(\mathbf{x}_{i,k-1} \leftarrow \mathbf{x}_{i,k})}{p(\bar{\mathbf{x}}_{i,k})}. \tag{1.16}$$

Here, the denominator is a joint pdf of the individual path vertices, i.e. $p(\bar{\mathbf{x}}_k) = p_A(\mathbf{x}_0, \cdots, \mathbf{x}_k)$. To calculate this, let $\mathcal{S}$ be the set of surfaces in the scene and $A(S)$ be the surface area of surface $S \in \mathcal{S}$. Then the probability of selecting S is $A(S)/\sum_{S' \in \mathcal{S}} A(S')$ and the probability of selecting a point from S is $1/A(S)$, so

$$p_A(\mathbf{x}_i) = \frac{1}{\sum_{S \in \mathcal{S}} A(S)}. \tag{1.17}$$

Hence, every point has an equal probability to be sampled for each path vertex, and each path vertex has an equal weight of integration. This in turn implies that path vertices are independent of each other in the area domain, simplifying our path probability to $p(\bar{\mathbf{x}}_k) = \prod_{i=1}^{k} p_A(\mathbf{x}_i)$. We then choose $W_e^j(\cdot)$ such that it contains the $G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)/p_A(\mathbf{x}_1)$ term, further simplifying matters. By folding the probabilities into the throughput, we derive the practical throughput function

$$\beta_A(\bar{\mathbf{x}}_k) = \frac{T(\bar{\mathbf{x}}_k)}{\prod_{i=1}^{k-1} p_A(\mathbf{x}_{i+1})}. \tag{1.18}$$

In turn, we can simplify our previous formulation of $\langle I_j \rangle_{MC}$ to

$$\langle I_j \rangle_{MC} = \frac{1}{N} \sum_{i=1}^{N} W_e^j(\mathbf{x}_{i,0}, \mathbf{x}_{i,1}) \beta_A(\bar{\mathbf{x}}_{i,k}) L_e(\mathbf{x}_{i,k-1} \leftarrow \mathbf{x}_{i,k}). \tag{1.19}$$

Unfortunately, although having every surface point be equally weighted simplifies the Monte Carlo estimator, it means that any surface point can be chosen, including those that don't make sense. For example, Figure 1.3 describes two completely disconnected rooms where it would be physically impossible to reach $\mathbf{y}$ from $\mathbf{x}$, yet the area domain sampling strategy allows for this since $p_A(\mathbf{y}) > 0$. This means that $\langle I_j \rangle_{MC}$ will have inherently higher variance, since we introduce zero-contribution samples for a clearly nonzero integral. Furthermore, every single

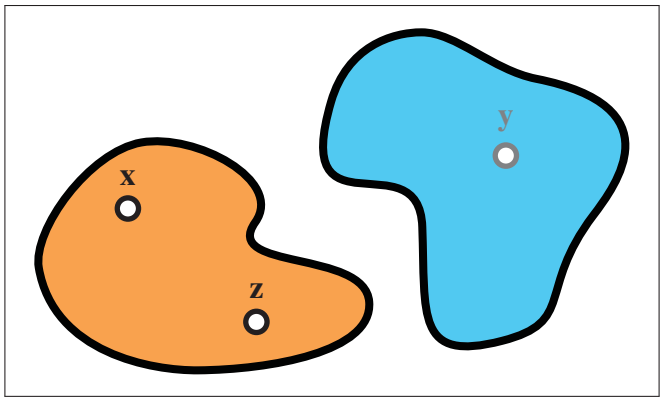


Figure 1.3 Despite being in separate disconnected rooms (orange and blue), **y** and **z** can both be sampled from **x** in an area-domain sampling strategy. However, with solid angle sampling it is impossible to select **y** from **x**, thus preserving physical limitations

sample requires k evaluations of the visibility term, which tends to be expensive even on modern systems (Bartels & Harada, 2022).

1.3.1 Local sampling for practical path construction

To solve these issues, we revert back to locally sampling in the solid angle domain as originally introduced in Equation (1.1). In doing so, we incrementally construct the path vertex-by-vertex starting from the sensor vertex, making sure that the domain of possible vertices $\mathbf{x}_{i+1}$ is limited to only those that are visible from the current vertex $\mathbf{x}_i$. For example, starting from $\mathbf{x}_1$ in Figure 1.1, we would sample $\mathbf{x}_2$, then from $\mathbf{x}_2$ we would sample $\mathbf{x}_3$, and so on. This is preferable as it allows us to avoid directly sampling occluded vertices such as $\mathbf{x}'_3$ from $\mathbf{x}_2$, which in turn significantly reduces the number of zero-contribution samples.

We start by noting that $f_r(\mathbf{x}_{i+1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i-1}) = f_r(\mathbf{x}_i, \omega_{\mathbf{x}_i \rightarrow \mathbf{x}_{i-1}}, \omega_{\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}})$, giving us a means to write the BSDF in terms of solid angle directions rather than path vertices. Converting $p(\bar{\mathbf{x}}_k)$ to solid angle measure then gives

$$p_A(\mathbf{x}_{i+1}) = p_\Omega(\omega_{\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}}) \left| \frac{\mathrm{d}\omega}{\mathrm{d}A} \right| = p_\Omega(\omega_{\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}}) \frac{\left| \mathbf{n}_{\mathbf{x}_{i+1}} \cdot \omega_{\mathbf{x}_{i+1} \rightarrow \mathbf{x}_i} \right|}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2}, \tag{1.20}$$

where $|\mathrm{d}\omega/\mathrm{d}A| = \frac{|\mathbf{n}_\mathbf{y}\cdot\omega|}{r^2}$ is the Jacobian of the transformation mapping $\mathbf{x}_{i+1}$ to $\omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}}$ (Pharr *et al.*, 2017). Since paths are constructed incrementally, the solid angle pdf is actually a conditional pdf on all previous vertices in the path, i.e.

$$p_\Omega(\omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}} \mid \mathbf{x}_1, \cdots, \mathbf{x}_i) = p_A(\mathbf{x}_{i+1})\frac{\|\mathbf{x}_{i+1}-\mathbf{x}_i\|^2}{\left|\mathbf{n}_{\mathbf{x}_{i+1}}\cdot\omega_{\mathbf{x}_{i+1}\to\mathbf{x}_i}\right|}. \tag{1.21}$$

Note that owing to the chain rule, multiplying these conditional pdfs together recovers the original path distribution:

$$p(\bar{\mathbf{x}}_k) = p_\Omega(\omega_{\mathbf{x}_0\to\mathbf{x}_1})p_\Omega(\omega_{\mathbf{x}_1\to\mathbf{x}_2} \mid \mathbf{x}_1)\cdots p_A(\omega_{\mathbf{x}_{k-1}\to\mathbf{x}_k} \mid \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{k-1}). \tag{1.22}$$

This allows us to reuse our earlier definitions of the throughput function. Substituting Equation (1.20) into Equations (1.13) and (1.18) allows us to almost entirely cancel out $G(\mathbf{x}_{i+1}\leftrightarrow\mathbf{x}_i)$, which gives the updated solid angle formulations:

$$T(\bar{\mathbf{x}}_k) = \prod_{i=1}^{k-1} f_r(\mathbf{x}_i, \omega_{\mathbf{x}_i\to\mathbf{x}_{i-1}}, \omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}})\left|\mathbf{n}_{\mathbf{x}_i}\cdot\omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}}\right|, \tag{1.23}$$

$$\beta(\bar{\mathbf{x}}_k) = \frac{T(\bar{\mathbf{x}}_k)}{\prod_{i=1}^{k-1} p_\Omega(\omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}} \mid \mathbf{x}_1, \cdots, \mathbf{x}_i)}. \tag{1.24}$$

Further substitution into Equation (1.19) gives the solid angle estimator

$$\langle I_j\rangle_{MC} = \frac{1}{N}\sum_{i=1}^{N} W_e^j(\mathbf{x}_{i,0}, \mathbf{x}_{i,1})\beta(\bar{\mathbf{x}}_{i,k})L_e(\mathbf{x}_k, \omega_{\mathbf{x}_k\to\mathbf{x}_{k-1}}). \tag{1.25}$$

For convenience, from this point forward we assume pdfs are in the solid-angle domain unless otherwise specified, i.e. $p = p_\Omega$. We also replace $\omega_{\mathbf{x}_i\to\mathbf{x}_{i-1}}$ with ω_o and $\omega_{\mathbf{x}_i\to\mathbf{x}_{i+1}}$ with ω_i for consistency with the variables originally introduced in Equation (1.1).

Sampling directions ω_i is easy enough; we can uniformly sample over the hemisphere H^2 centred at $\mathbf{n}_{\mathbf{x}_i}$, with corresponding pdf $p(\omega_i) = \frac{1}{2\pi}$. However, we still need to generate the next vertex $\mathbf{x}_{i+1}$ from our sampled ω_i. Hence, we introduce the concept of the closest-hit intersection

function `Intersect`$(\mathbf{x}, \omega_i)$, which returns the closest surface point to $\mathbf{x}$ that lies on the ray with origin $\mathbf{x}$ and direction ω_i. Algorithm 1.1 provides a simple path tracing implementation, demonstrating how $\beta(\bar{\mathbf{x}}_k)$ is iteratively built until path termination.

Algorithm 1.1 Path tracing

```
1  function TracePath(x_0, ω_i, k)
2      L ← 0
3      β ← 1
4      x_i ← x_0
5      while path length is less than k do
6          x_i ← Intersect(x_i, ω_i)
7          ω_o ← −ω_i
8          ω_i ←new sampled direction from x_i
9          L ← L + β · L_e(x_i, ω_o)
10         β ← β · f_r(x_i, ω_o, ω_i)|n_{x_i} · ω_i| / p(ω_i)
11     end while
12     return L
13 end
```

1.4 Variance reduction

Although path tracing provides an unbiased estimator for the rendering equation, in its most basic form it also tends to be very noisy. We can improve the quality of the rendered image by increasing the number of path samples, but MC integration's $O(1/\sqrt{N})$ convergence rate makes this a slow process. Instead, we look towards reducing the variance of the MC estimator itself. Recall from Equation (1.10) that $\text{Var}(\langle I \rangle_{MC}) = \frac{1}{N} \sum_{i=1}^{N} \text{Var}\left(\frac{f(X_i)}{p(X_i)}\right)$; hence, variance increases if f and p are dissimilar across samples X_i. This noise can be attributed primarily to two sources: the first being the visibility term, and the second being discrepancies between the realized path sampling distribution $p(\bar{\mathbf{x}})$ and the integrand $W_e^j T L_e$. Since solid-angle sampling mitigates the former issue, practical variance reduction focuses on solving the latter.

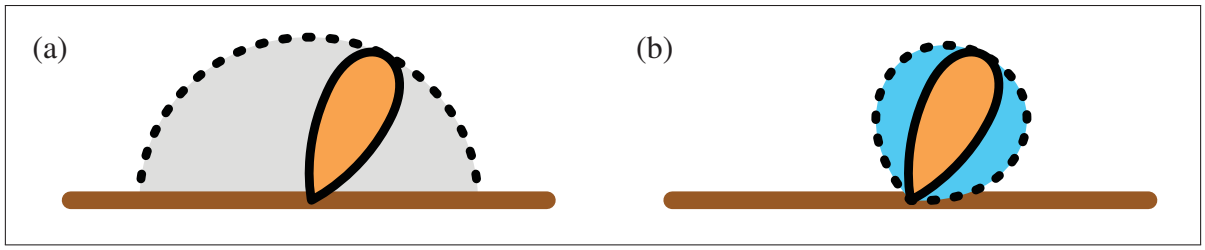


Figure 1.4 **(a)** A uniform distribution can differ significantly from the integrand, which increases variance. **(b)** To counteract this, importance sampling uses a distribution that closely matches the integrand, allowing f/p to be more consistent across samples

1.4.1 Importance sampling

If we choose $p \propto f$, we are able to achieve a zero-variance Monte Carlo estimator. Indeed, the ideal scenario arises when we set $p = f/I$; unfortunately, since it depends on knowing the true value of I prior to integration, this approach is circular. However, if we choose a sampling distribution p that is similar to f as in Figure 1.4b, then f/p is made consistent, reducing the variance of the entire estimator. This approach is known as *importance sampling* (Owen, 2013), and forms the crux of most of the variance reduction methods used in rendering today.

When applying importance sampling in the context of path tracing, an obvious target value for f is the throughput function T from Equation (1.23). Although we would ideally want to importance sample the entire path $\bar{\mathbf{x}}$, this is infeasible since the path distribution is usually intractable. Instead, examining each term in the expansion of $T(\bar{\mathbf{x}})$ suggests that at every path vertex $\mathbf{x}_i$, we should try to importance sample $f_r(\mathbf{x}_i, \omega_o, \omega_i)|\mathbf{n}_\mathbf{x} \cdot \omega_i|$. Alternatively, the formulation in Equation (1.14) suggests that we can directly importance sample the path contribution by choosing $\mathbf{x}_k$ such that $L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)$ is nonzero. These two strategies are exclusive, meaning that we must choose one or the other for sampling a path vertex; we discuss their advantages and drawbacks below.

BSDF importance sampling. Recall that BSDFs are themselves statistical distributions; specifically, we can view a BSDF as modelling the directional distribution $p(\omega_o)$ of outgoing rays with origin $\mathbf{x}_i$ given an incoming light direction ω_i and vice-versa, allowing us to perfectly importance sample f_r. Although glossy and microfacet BSDFs are usually impossible to sample

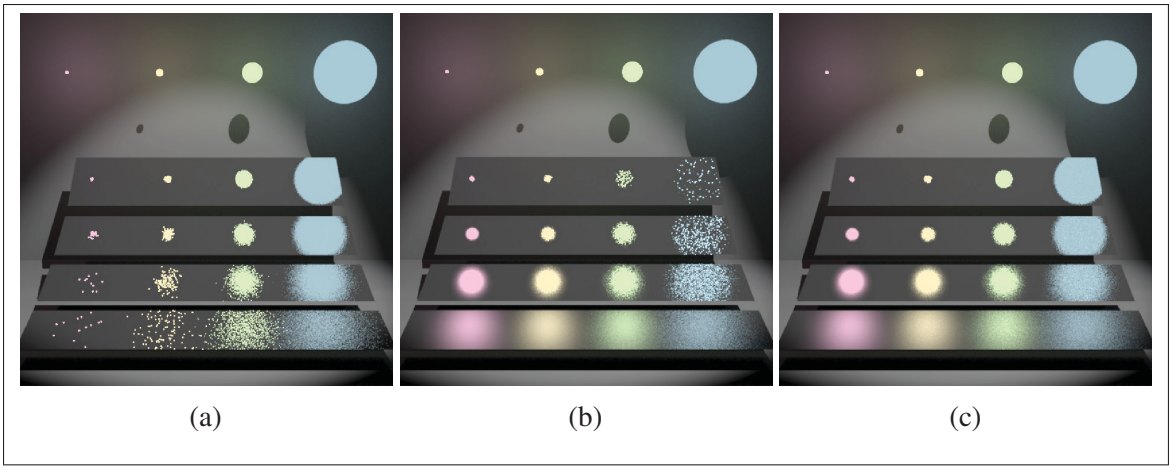


Figure 1.5 Comparing the results of **(a)** BSDF sampling, **(b)** NEE sampling, and **(c)** MIS between the two strategies Taken from Veach (1997, chapter 9)

from analytically, there exist many approaches for sampling distributions that are close to these BSDFs (Lawrence, Rusinkiewicz & Ramamoorthi, 2004; Heitz & d'Eon, 2014; Heitz *et al.*, 2016; Chermain, Sauvage, Dischler & Dachsbacher, 2021). However, a key drawback to BSDF importance sampling is that the emissive term is not considered, meaning it is quite possible to generate a zero-contribution path. Figure 1.5a shows how importance sampling diffuse surfaces will often fail to account for small light sources, since there is a very low probability that the outgoing ray will encounter the light, leading to a noisy result.

Next-event estimation. To mitigate the problem of generating zero-contribution paths, we can choose to sample the terminal vertex $\mathbf{x}_k$ over light emitters in the scene with an area sampling strategy; this process is known as next-event estimation (NEE). In the naive case, we sample uniformly over all emissive surfaces $\mathcal{S}_{\text{emissive}}$, with probability

$$p_{\text{emissive}} = \frac{1}{\sum_{S \in \mathcal{S}_{\text{emissive}}} A(S)}. \tag{1.26}$$

This in turn gives the NEE estimator

$$\langle I_j \rangle_{NEE} = \frac{1}{N} \sum_{i=1}^{N} W_e^j(\mathbf{x}_{i,0}, \mathbf{x}_{i,1}) \beta(\bar{\mathbf{x}}_{i,k-1}) \frac{G(\mathbf{x}_{i,k-1} \leftrightarrow \mathbf{x}_{i,k}) L_e(\mathbf{x}_{i,k-1} \leftarrow \mathbf{x}_{i,k})}{p_{\text{emissive}}}. \tag{1.27}$$

A better option, however, is to importance sample lights based on their intensity, and then uniformly sample over their area. If we sample emitters $\mathcal{L}_i \sim q$ for some pdf q, we derive the NEE estimator

$$\langle I_j \rangle_{NEE} = \frac{1}{N} \sum_{i=1}^{N} W_e^j(\mathbf{x}_{i,0}, \mathbf{x}_{i,1}) \beta(\bar{\mathbf{x}}_{i,k-1}) \frac{G(\mathbf{x}_{i,k-1} \leftrightarrow \mathbf{x}_{i,k}) L_e(\mathbf{x}_{i,k-1} \leftarrow \mathbf{x}_{i,k})}{p_A(\mathbf{x}_{i,k} \mid \mathcal{L}_i) q(\mathcal{L}_i)}. \tag{1.28}$$

Unfortunately, though we now have a much higher likelihood of sampling contributing paths, the extra light sampling step tends to introduce additional noise into the equation. Furthermore, next event estimation on highly specular surfaces frequently fails due to the narrowness of the specular lobe causing $T(\bar{\mathbf{x}})$ to be near-zero. Figure 1.5b demonstrates this, where the reflection of the large sphere on the top, most specular panel is extremely noisy.

1.4.2 Multiple importance sampling

Despite being an improvement over uniform sampling, importance sampling still fails to cover complex product distributions. For example, BSDF and NEE sampling each partially sample the path throughput, with neither method able to fully sample the entire path distribution. Veach & Guibas (1995) thus propose *multiple importance sampling* (MIS) as a means of overcoming this limitation. Figure 1.6a demonstrates this process: by introducing a weight term for each available sampling strategy, we mix their contributions and generate a more efficient estimator. Generally, suppose we have a set of candidate distributions $(p_1, p_2, \cdots, p_M)$; then

$$\langle I \rangle_{MIS} = \sum_{i=1}^{M} m_i(X_i) \frac{f(X_i)}{p_i(X_i)} \quad X_i \sim p_i(x), \tag{1.29}$$

provides an unbiased estimator for I given that (1) $\sum_{i=1}^{M} m_i(x) = 1$ for any $x \in \bigcup_{i=1}^{M} \text{supp}(X_i)$, and (2) $m_i(x) = 0$ if $x \notin \text{supp}(X_i)$. Similarly to Monte Carlo integration, it is imperative that $\bigcup_{i=1}^{M} \text{supp}(X_i)$ covers f, otherwise the final estimator will be biased.

An important, albeit difficult, factor when performing MIS is choosing the weights $m_i(X_i)$. One choice are the naive weights $m_i(x) = 1/M$; being constant, they satisfy the first condition, but fail the second if any of the candidate distributions do not cover f. In such cases, we may wish to use more advanced weights such as the balance heuristic:

$$m_i(x) = \frac{p_i(x)}{\sum_{j=1}^{M} p_j(x)}. \tag{1.30}$$

Variance-wise, the balance heuristic also improves upon naive weights, and in fact is provably optimal (Veach & Guibas, 1995, Theorem 1). The power heuristic is a variant of the balance heuristic that raises all the p_i to a given constant β, and is suitable for applications where one of the p_i is already a close match for the integrand f:

$$m_i(x) = \frac{p_i(x)^\beta}{\sum_{j=1}^{M} p_j(x)^\beta}. \tag{1.31}$$

Unfortunately, evaluating Equation (1.29) with the above balance heuristics has $O(M^2)$ cost, which quickly becomes expensive in path tracing. This is problematic if we are required to evaluate $m_i(x)$ in a tight loop with a large number of sampling strategies. Furthermore, the variance of an MIS estimator cannot be less than the individual variances of the Monte Carlo estimators corresponding to each of the candidate distributions. This means that if one of the constituent sampling strategies is provably optimal, MIS will always give a worse result. However, this is counterbalanced by the fact that MIS allows us to mix between many different sampling strategies which together cover the ideal sampling distribution, thus deriving a simple yet robust estimator.

MIS is especially useful in path tracing, since neither BSDF nor light importance sampling are able to capture the complete path distribution; in fact, the scenarios where one strategy performs

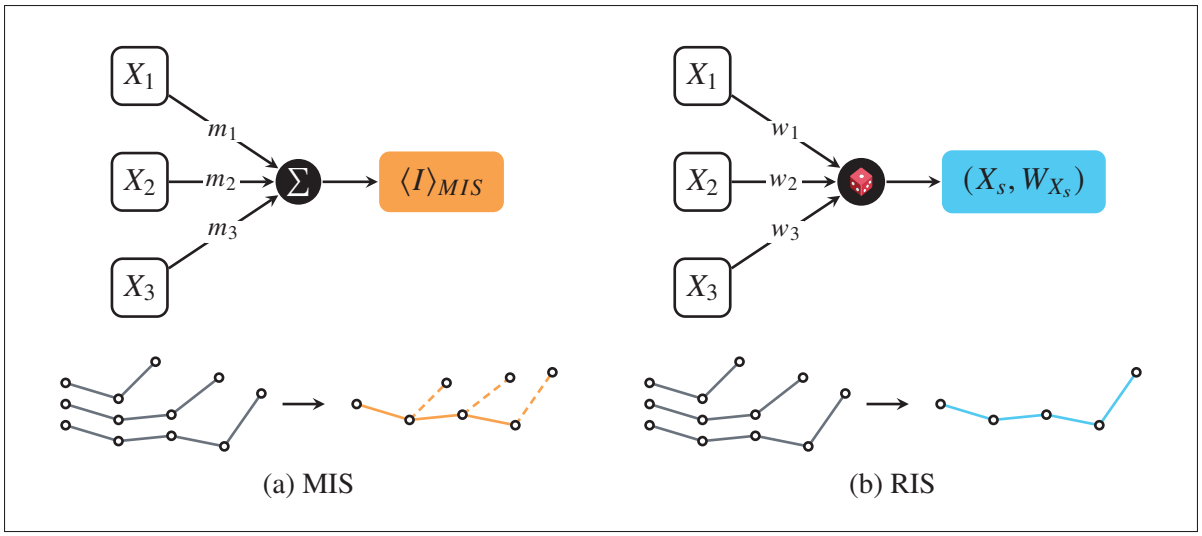


Figure 1.6 **(a)** MIS weighs the contributions of the candidate samples to output an efficient estimator. **(b)** On the contrary, RIS selects one of the candidate samples, eventually producing a sample pool with distribution equal to the target function

well tend to be areas of weakness for the other. With MIS, we can sample a path vertex with each strategy and combine their contributions to get the best of both worlds, as Figure 1.5c shows. MIS can also be extended over a continuous technique space such as what might be considered in path reuse or volume scattering, further enhancing its flexibility (West, Georgiev, Gruson & Hachisuka, 2020).

1.4.3 Resampled importance sampling

Resampled importance sampling (RIS) offers another solution for sampling from a complex distribution; as its name suggests, it outputs a single sample *resampled* from a pool of already-generated samples (Talbot *et al.*, 2005). Suppose we have a distribution $\hat{p}(x)$ that we wish to sample from, but which is expensive to evaluate or cannot be efficiently sampled. Given a pool of candidate samples $(X_1, X_2, \cdots, X_M)$ with respective distributions $p_1(x), p_2(x), \cdots, p_M(x)$, RIS will return an index s such that X_s has asymptotic sampling distribution $\hat{p}(x)$. We define the corresponding candidate RIS weights as

$$w_i = m_i(X_i)\hat{p}(X_i)W_{X_i}, \tag{1.32}$$

where $m_i(x)$ is the MIS weight corresponding to $p_i(x)$, $\hat{p}(x)$ denotes a user-defined *target function* such that $\hat{p}(x) > 0$ wherever $f(x) \neq 0$, and W_{X_i} is an unbiased contribution weight (UCW) for X_i. Note that if $p_i(x)$ is known, then $W_{X_i} = 1/p_i(X_i)$; this helps simplify the formulation in practice. We then choose X_s with probability $\frac{w_s}{\sum_{i=1}^{M} w_i}$ and calculate its UCW:

$$W_{X_s} = \frac{1}{\hat{p}(X_s)} \sum_{i=1}^{M} w_i. \tag{1.33}$$

Hence, RIS returns both the selected sample and a corresponding UCW, as Figure 1.6b demonstrates. To finish the RIS process, we let $Y = X_s$, giving an unbiased estimator for I:

$$\langle I \rangle_{RIS} = f(Y) W_Y. \tag{1.34}$$

The target function $\hat{p}(x)$ is especially important to the theory of RIS. For one, the realized sampling distribution $\bar{p}(x)$ of X eventually converges to $\hat{p}(x)$, so that setting $\hat{p}(x) = f(x)$ allows us to directly estimate the latter.[1] However, if $f(x)$ is expensive to evaluate, we can use $\hat{p}(x)$ as a cheaper proxy for $f(x)$ in calculating our weights at no additional cost. Furthermore, W_X proxies $1/\bar{p}(X)$, and can be substituted for the latter without having to evaluate a potentially expensive pdf.

Like MIS, RIS allows us to efficiently estimate $\int f(x)\mathrm{d}x$ when $f(x)$ is not a practical distribution to sample from. Also like MIS, the supports of the constituent distributions in RIS must together cover that of f; $\langle I \rangle_{RIS}$ is unbiased only if there exists an index $1 \leq i \leq M$ such that $p_i(x) > 0$ when $f(x) \neq 0$. However, where MIS directly produces an efficient estimator for f, RIS produces a sample pool with distribution proportional to f, which is useful in situations where sample generation from a complex distribution is desired (Zheng & Zwicker, 2019; Sugimoto, Chen, Jiang, Batty & Hachisuka, 2023). In path tracing, this allows us to return path samples directly, permitting their use in downstream passes. Because of this, RIS can also be chained by feeding the produced samples into another RIS process. The original UCW formulation

[1] Assuming $\hat{p}(x)$ is normalized; if not, then $\bar{p}(x) \to \hat{p}(x)/\int \hat{p}(x)\mathrm{d}x$.

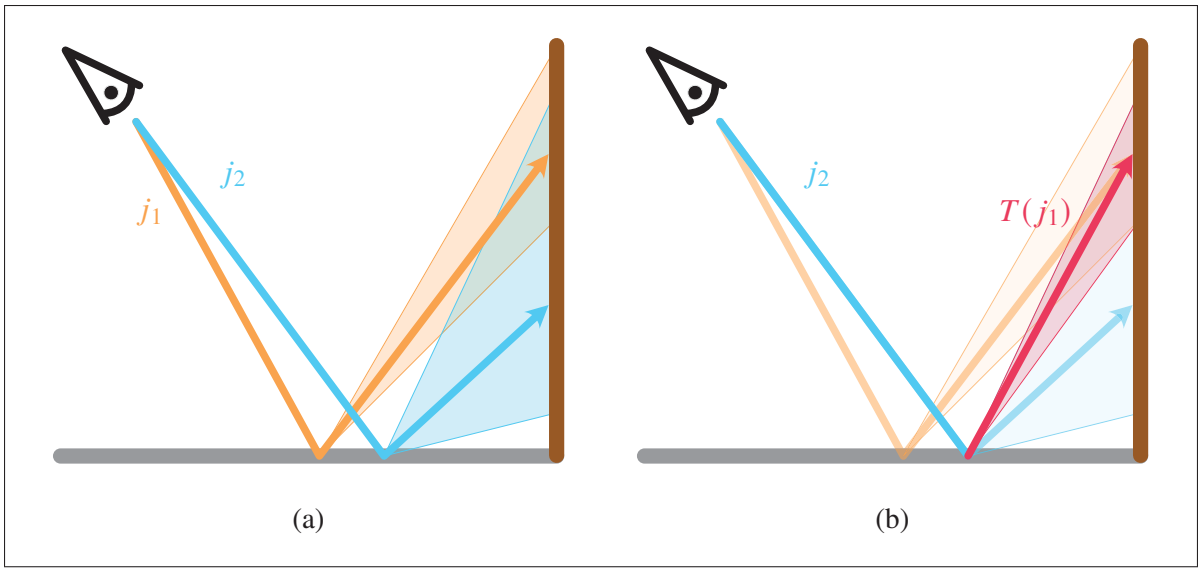


Figure 1.7 **(a)** The path space for pixel j_1 varies from that for pixel j_2 as their primary hit points differ in position, which in turn influences the domain of the sampled secondary ray. **(b)** Shift mappings allow us to evaluate paths from j_1 in the domain of j_2, allowing for reuse

with known p_i is then replaced by the UCW calculated in Equation (1.33), allowing for iterative refinement of the sample pool.

So far, we have assumed that the input samples X_i share the same domain. However, in rendering we often seek to evaluate paths in domains different from the one they were sampled in; Figure 1.7 illustrates how the path space between pixels j_1 and j_2 can differ significantly. Gradient-domain rendering (Kettunen *et al.*, 2015) introduces the concept of *shift mappings*, which allows for mapping samples from one pixel domain to the other as long as they overlap.

1.5 Shift mappings

Mathematically, a shift mapping is a function $T : \Omega \to \Psi$ that allows us to reuse samples $x \in \Omega$ as $y = T(x) \in \Psi$. Crucially, there must also exist a corresponding inverse mapping $T^{-1} : \Psi \to \Omega$, so that $x = T^{-1}(y)$, and if we wish to evaluate the pdf of the shifted samples, T must have a Jacobian (Hua *et al.*, 2019). We can also think of shift mappings as changing

the density of the shifted samples compared to the base samples, and this scaling factor is represented by the determinant of the Jacobian, $|T'|$.

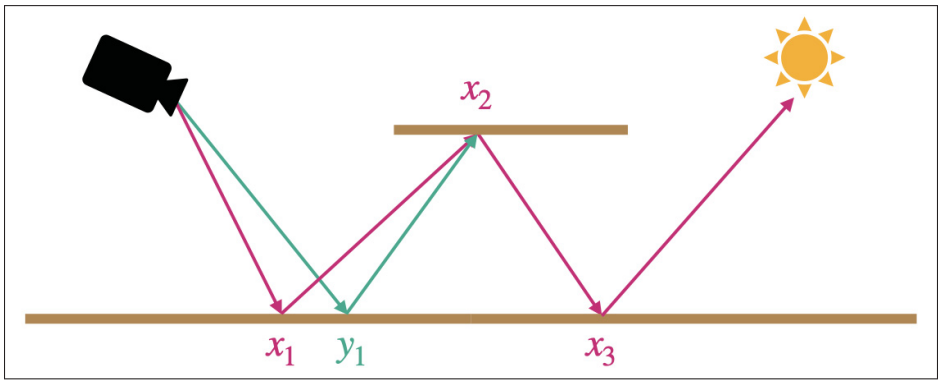


Figure 1.8 Reconnection shift mapping

The simplest shift mapping for paths is reconnection shift mapping, which simply connects two paths at a given vertex (Hua *et al.*, 2019). Figure 1.8 demonstrates this on paths $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots]$ and $\bar{\mathbf{y}} = [\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \cdots]$, where we can shift the *base path* $\bar{\mathbf{x}}$ onto $\bar{\mathbf{y}}$ at $\mathbf{x}_2$ to create the new *offset path* $\bar{\mathbf{x}}' = [\mathbf{y}_0, \mathbf{y}_1, \mathbf{x}_2, \cdots]$. Kettunen *et al.* (2015) gives us the Jacobian of the reconnection shift in the area domain as 1; hence, if we take $\mathbf{y}_{i+1} = \mathbf{x}_{i+1}$ as the reconnection vertex and let $\omega_i^{\mathbf{y}} = \omega_{\mathbf{y}_i \to \mathbf{y}_{i+1}}$ be the incident direction at $\mathbf{y}_i$ (and likewise with $\omega_i^{\mathbf{x}}$ for $\mathbf{x}_i$), applying Equation (1.20) gives the solid-angle Jacobian as

$$\left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}A}\right|\left|\frac{\mathrm{d}A}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \left|\frac{\mathbf{n}_{\mathbf{y}_{i+1}} \cdot \omega_i^{\mathbf{y}}}{\mathbf{n}_{\mathbf{x}_{i+1}} \cdot \omega_i^{\mathbf{x}}}\right| \frac{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|^2}{\|\mathbf{y}_i - \mathbf{y}_{i+1}\|^2}. \tag{1.35}$$

The Jacobian of the offset path is then the product of the Jacobians at the individual vertices:

$$\left|\frac{\mathrm{d}\bar{\mathbf{x}}'}{\mathrm{d}\bar{\mathbf{x}}}\right| = \prod_{j=1}^{k} \left|\frac{\mathrm{d}\omega_i^{\mathbf{x}'_j}}{\mathrm{d}\omega_i^{\mathbf{x}_j}}\right|. \tag{1.36}$$

This implies that we can calculate individual Jacobian terms during local sampling of each vertex rather than trying to construct an unwieldy Jacobian for the entire path at once, which greatly simplifies implementing shift mappings.

Shift mappings are essential for path reuse, since they allow reusing some or all of the base path within the another pixel's path domain. However, they can fail if the shifted sample y does not fall within the desired target domain, $p(y)$ is very small or very large, or the Jacobian is indeterminate. For example, if y_1 lies on a specular surface, then attempting to reconnect it to x_2 may lead to an infinitesimal pdf. In these cases, the offset path is invalid, which forces a new path to be sampled in the target domain and reduces the efficiency of path reuse.

1.6 Real-time global illumination

In real-time contexts such as video games, additional timing constraints must be imposed upon the rendering loop in order to maintain a reasonable degree of interactivity and responsiveness. The typical frame rate target of 60 Hz leaves only 16 milliseconds to compute physics simulations, resource bindings, and API overheads for each frame on top of actual rendering tasks (Gregory, 2026, chapter 11); this results in a limited time budget for the latter, which has led to numerous approximations for rendering expensive indirect illumination contributions. Baked global illumination methods precompute the indirect contribution and store it in a special data structure to be cheaply queried at render time (Luksch, Wimmer & Schwärzler, 2019); these include techniques such as reflective shadow maps (Dachsbacher & Stamminger, 2005), spherical harmonic lighting (Green, 2003), and many-light techniques (Dachsbacher *et al.*, 2014). Such baking precludes lighting that moves or changes colour and intensity, so for real-time dynamic illumination hierarchical algorithms are used to limit the amount of computation required (Crassin, Neyret, Sainz, Green & Eisemann, 2011; Majercik, Guertin, Nowrouzezahrai & McGuire, 2019; Lin & Yuksel, 2020; Tokuyoshi, Ikeda, Kulkarni & Harada, 2024). However, being approximations, these techniques do not accurately model real-world lighting, leaving much to be desired.

Monte Carlo path tracing solves the accuracy problem; unfortunately, the high sample counts required for convergence make it infeasible for real-time rendering. Most current efforts focus on reducing the number of samples required to generate a usable image via adaptive sampling, denoising, or super-resolution algorithms (Bálint, Salaün, Seidel & Myszkowski, 2026), but

these are still heavily reliant on the quality of their input samples. One can choose to raise the quality of the path samples through path guiding (Lu, Chang, Hedstrom & Li, 2024; Alber, Hanika & Dachsbacher, 2025) and Markov Chain Monte Carlo methods (Dittebrandt, Schüßler, Hanika, Herholz & Dachsbacher, 2023; Holl, Singh & Seidel, 2026). Alternatively, sample reuse allows for one sample to have the power of previously computed samples, drastically reducing the number of samples required to produce a usable image (Dittebrandt, Hanika & Dachsbacher, 2020). We focus our efforts on ReSTIR, the sample reuse method proposed by Bitterli *et al.* (2020) which has seen much practical success.

CHAPTER 2

RESERVOIR-BASED SPATIOTEMPORAL IMPORTANCE RESAMPLING

2.1 Introduction

The ultimate goal of real-time path tracing is to perform unbiased, low-variance Monte Carlo integration of the rendering equation. However, a convergent result requires a large number of samples, and existing variance reduction methods are not efficient enough for real-time rendering. Bitterli *et al.* (2020) achieve this by introducing ReSTIR, which combines resampled importance sampling with reservoir sampling to reuse samples taken from past frames and surrounding pixels. RIS serves as a foundational method to produce samples that closely follows the sampling distribution of the rendering equation's integrand, while reservoir sampling replaces RIS's requirement of an expensive linear-space input sample pool with a constant-space streaming algorithm. Because RIS can be easily chained, this process results in an estimator where each sample contains information from all previous RIS iterations, amplifying the effective sample size and drastically reducing variance. Instead of the large number of samples typically required for convergence, ReSTIR only needs one or two samples per frame to produce a highly converged image, well within the real-time capabilities of current-generation graphics hardware.

In this chapter, we summarize the introductory work of Wyman *et al.* (2023) on ReSTIR. We first examine its theoretical background, particularly the use of reservoir sampling and the application of shift mappings to enable sample reuse across domains. We then describe practical considerations for implementing ReSTIR that are critical for ensuring its efficiency and unbiasedness. We further review several substantially impactful variants of the base algorithm as well as state-of-the-art improvements. Finally, we discuss known drawbacks of ReSTIR and their possible solutions.

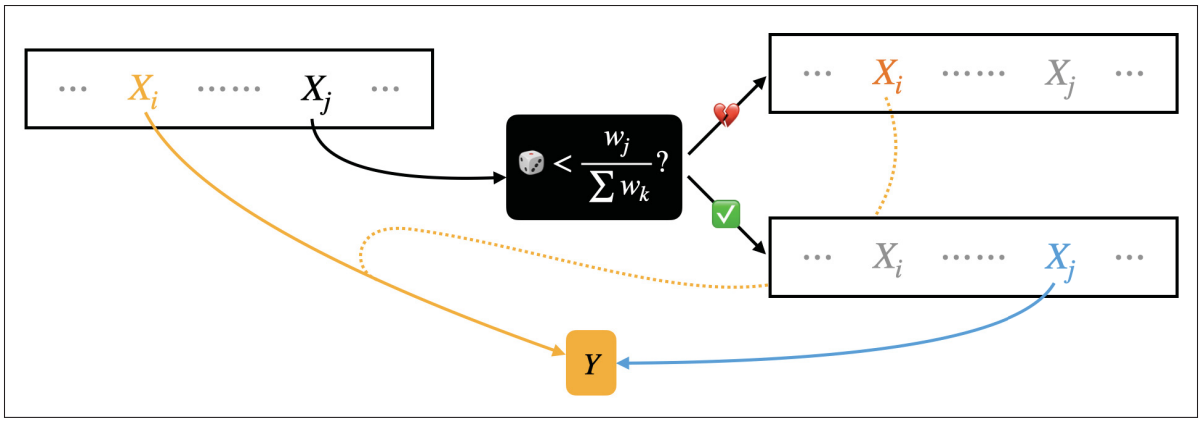


Figure 2.1 Visualization of updating the reservoir in WRS. Here, the reservoir Y has chosen X_i, and we are testing if X_j should replace it. Replacement only succeeds if $\text{Unif}[0, 1] < \frac{w_j}{\sum w_k}$, otherwise X_i remains in the reservoir

2.2 Method

The core idea of ReSTIR is that we can use RIS's chaining ability to sample from surrounding pixels and previous frames, without having to spend additional memory or compute cycles to store and preprocess the sample pool. This process requires two steps: *weighted reservoir sampling* to importance sample from a stream of samples or reservoirs, and *reservoir resampling* to define the source reservoirs and their sampling weights.

2.2.1 Weighted reservoir sampling

Reservoir sampling is a method for uniformly selecting M samples from a stream of candidate samples $\{x_1, x_2, \cdots, x_N\}$ in a single pass and with constant memory cost (Vitter, 1985). It does so by continuously updating a special data structure called the *reservoir*, always ensuring that the M samples held within are selected uniformly; at the end of the sampling routine, the reservoir is returned with the assurance that each sample has been sampled with probability M/N. Going forward, we focus on the case where $M = 1$; in other words, the reservoir stores a single sample.

Weighted reservoir sampling (WRS) extends reservoir sampling by associating a weight w_i with each candidate x_i in the stream, thus allowing us to sample according to any desired distribution

(Chao, 1982). In WRS, sample x_i is thus selected with probability $w_i / \sum_{j=1}^{N} w_j$. Hence, uniform reservoir sampling can be seen as a special case of WRS where $w_i = 1$ for all i. We visualize the WRS process in Figure 2.1. We also prove that the iterative nature of WRS returns a sample that follows the desired sampling distribution in Appendix C.

Applying WRS to RIS. Resampling from the candidate pool in RIS offers a very natural application of WRS and allows us to avoid storing every candidate sample, providing large benefits in memory-constrained GPU environments. To do so, we store the UCW in the reservoir, as seen in Algorithm 2.1. We then supply the RIS weights to the reservoir update process, before finally setting the UCW of the output reservoir according to Equation (1.33). The output sample can then be reinserted into the source sample pool to implement RIS chaining.

Algorithm 2.1 Reservoir RIS

```
1  struct Reservoir
2      Y ← ∅
3      W ← 0
4      w_sum ← 0
       // Attempt to replace the probability of the stored sample
       // W is stored after all candidate samples have been read
5      function update(x_i, w_i)
6          w_sum ← w_sum + w_i
7          u ← Random()
8          if u < w_i / w_sum then
9              Y ← x_i
10         end if
11     end
12 end
13 function ReservoirRIS(X := {x_1, x_2, ··· , x_n})
14     Reservoir r
15     foreach x_i ∈ X do
16         w_i ← m_i(x_i) p̂(x_i) W_{x_i}                    // Equation (1.32)
17         r.update(x_i, w_i)
18     end foreach
19     r.W ← 1 / p̂(r.Y) r.w_sum                             // Equation (1.33)
20     return r
21 end
```

Reservoir merging. In addition to being performed on streams of samples, the combined WRS and RIS process can also be performed on streams of reservoirs, resulting in a process called *reservoir merging*. Assume we have two reservoirs r_1 and r_2 that we wish to merge; we then calculate the RIS weights of each reservoir's sample using the respective stored UCW weights, and update the output reservoir and final UCW as usual. Algorithm 2.2 generalizes reservoir merging to a stream of n source reservoirs, in which case we perform the update process in a loop. As above, RIS chaining can be implemented with reservoir merging, though in a slightly simpler manner given the homogeneity of the input and output types.

Algorithm 2.2 Reservoir merging

```
function MergeReservoirs(r_1, r_2, ··· , r_n)
    Reservoir r
    for i ← 1 to n do
        calculate MIS weights m_i for each r_i
        r.update(r_i.Y, m_i(r_i.Y) p̂(r_i) r_i.W)
    end for
    r.W ← 1/p̂(r.Y) r.w_sum
    return r
end
```

2.2.2 Reservoir resampling

The implementation of RIS chaining with reservoir merging is known as *reservoir resampling*. Chaining allows for effective iterative refinement of the sample pool, allowing us to continually update the output reservoir with new information without needing to store all previous reservoirs or samples. Meanwhile, reservoir merging avoids having to recompute UCWs for the source samples, since these are stored in the source reservoirs. Reservoir resampling thus provides a compelling way to reuse path samples in rendering, where memory and time efficiency are prioritized.

Unfortunately, we must also consider that these source samples may come from completely different domains; for example, if the primary hit lies on a specular surface, which causes

Algorithm 2.3 Generalized MIS and RIS weights

```
   // Equation (2.2)
 1 function GeneralizedTargetFn(c, j, Reservoir r_j)
 2 |   x_j ← T_{c←j}(r_j.Y)
 3 |   J ← |T_{c←j}(r_j.Y)|
 4 |   if x_j ≠ 0 then
 5 |   |   return p̂_j(x_j) · J
 6 |   else
 7 |   |   return 0
 8 |   end if
 9 end
   // Equation (2.3)
10 function GeneralizedBalanceHeuristic(c, i, Reservoir r_i)
11 |   numer ← GeneralizedTargetFn(c, i, r_i)
12 |   denom ← numer
13 |   for j ← 1 to M; j ≠ i do
14 |   |   denom ← denom + GeneralizedTargetFn(c, j, r_j)
15 |   end for
16 |   return numer/denom
17 end
   // Equation (2.1)
18 function GeneralizedRISWeight(c, i, Reservoir r_i)
19 |   return GeneralizedBalanceHeuristic(c, i, r_i) · p̂(r_i.Y) · r_i.W · |T'_{c←i}(r_i.Y)|
20 end
```

surrounding pixels to have a completely different path space compared to the base pixel. Shift mappings as introduced in Section 1.5 allow us to shift all samples to a single pixel's domain when calculating the MIS weights, which we call the *canonical pixel*. Usually, the canonical pixel is intuitively chosen to be the current pixel that is being rendered. This also solves the requirement of unbiased RIS estimators that the union of the sampling distributions must cover the integrand, since this is true by definition for the path sampling distribution of the current pixel.

The introduction of shift mappings does require us to slightly modify our weight formulations. Letting c denote our canonical pixel with corresponding domain Ω, $T_{c\leftarrow i}(x) : \Omega_i \to \Omega$ denote the shift mapping from pixel i to c, and $Y_i = T_{c\leftarrow i}(X_i)$ be the shifted sample, we can rewrite

the RIS weight formulation from Equation (1.32) in terms of shift mappings into the canonical domain:

$$w_i = m_i(Y_i)\hat{p}(Y_i)W_{X_i}\left|T'_{c\leftarrow i}(X_i)\right|. \tag{2.1}$$

The Jacobian arises from the change of domain, and is especially important to prevent bias (Ouyang, Liu, Kettunen, Pharr & Pantaleoni, 2021). The formulations of $m_i(x)$ from Equations (1.30) and (1.31) must also take the shift mappings into account, as samples from Ω_i might have a different probability when shifted to Ω. We hence define a generalized target function for evaluating samples $y = T_{c\leftarrow i}(x)$ in the correct domain (Lin *et al.*, 2022), which proxies the probability of evaluating y in the canonical domain:

$$\hat{p}_{\leftarrow i}(y) = \begin{cases} \hat{p}_i(T_{i\leftarrow c}(y))\left|T'_{i\leftarrow c}(y)\right| & \text{if } y \in T_{c\leftarrow i}(\operatorname{supp} X_i), \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

This formulation is convenient because we can store $\hat{p}_i(x)$ in each source reservoir quite cheaply, meaning that evaluating $\hat{p}_{\leftarrow i}(y)$ is only as expensive as checking that the shift mapping succeeds and evaluating the Jacobian. Note that $T_{i\leftarrow c}(y) = T^{-1}_{c\leftarrow i}(y)$, so $\left|T'_{i\leftarrow c}(y)\right| = 1/\left|T'_{c\leftarrow i}(y)\right|$. Note also that if $c = i$, then $\hat{p}_{\leftarrow i}(y) = \hat{p}_c(y)$ since $T_{i\leftarrow c}$ is the identity function and its Jacobian is 1. We can now define the generalized balance heuristic by writing Equation (1.30) in terms of $\hat{p}(x)$:

$$m_i(y) = \frac{\hat{p}_{c\leftarrow i}(y)}{\sum_{j=1}^{M} \hat{p}_{c\leftarrow j}(y)}. \tag{2.3}$$

Since $\sum_{i=1}^{M} m_i(y) = 1$ as long as $y \in T_{c\leftarrow i}(\operatorname{supp} X_i)$, Equation (2.3) is a valid MIS weight and will produce an unbiased estimator. Algorithm 2.3 implements these generalized weights, completing the application of reservoir resampling to the rendering context.

2.3 ReSTIR in practice

The path tracing algorithm generates path samples which we can choose to store in reservoirs. If we choose to initialize each reservoir with one path sample, and assign a reservoir to each pixel

Algorithm 2.4 ReSTIR in practice

```
1  function Resample(r_1, r_2, ··· , r_M)
2      Reservoir r
3      for i ← 1 to M do
4          Y_i ← T_{c←i}(r_i.Y)                          // Shift samples from Ω_i to Ω_c
5          w_i ← GeneralizedRISWeight(c, i, r_i)
6          r.update(Y_i, w_i)                            // Same as Algorithm 2.2
7      end for
8      if r.Y ≠ ∅ then
9          r.W ← 1/p̂(r.Y) r.w_sum
10     end if
11     return r
12 end
```

within a single frame, we can then perform reservoir resampling on the combined reservoir pool, thus deriving the ReSTIR algorithm. Each returned reservoir contains a path sample which aggregates that pixel's path distribution; this reservoir is then added to the input reservoir pool for the next frame and the reservoir resampling process is repeated so that each path sample represents all previously seen samples, increasing the efficiency of the path tracing estimator and allowing for effective path reuse.

As hinted by its name, ReSTIR consists of reservoir resampling over both the spatial and temporal domain. Spatial reuse is the idea of resampling from surrounding pixels, and temporal reuse resamples the current motion-matched pixel from the previous frame. Combined, these comprise spatiotemporal reuse, and offer a particularly attractive application of reservoir resampling due to the large spatial sample pool motivating WRS and the long temporal sample chain motivating a chaining algorithm. Both forms of reuse base their effectiveness from the observation that spatially and temporally close pixels likely have very similar target functions, so samples from these pixels are almost perfectly importance sampled from each other's sampling distribution, providing a low-variance estimate. The bulk of variance reduction tends to come from temporal reuse, with spatial reuse helping mitigate artifacts that arise from overrepresented samples in the temporal chain (Sawhney *et al.*, 2023).

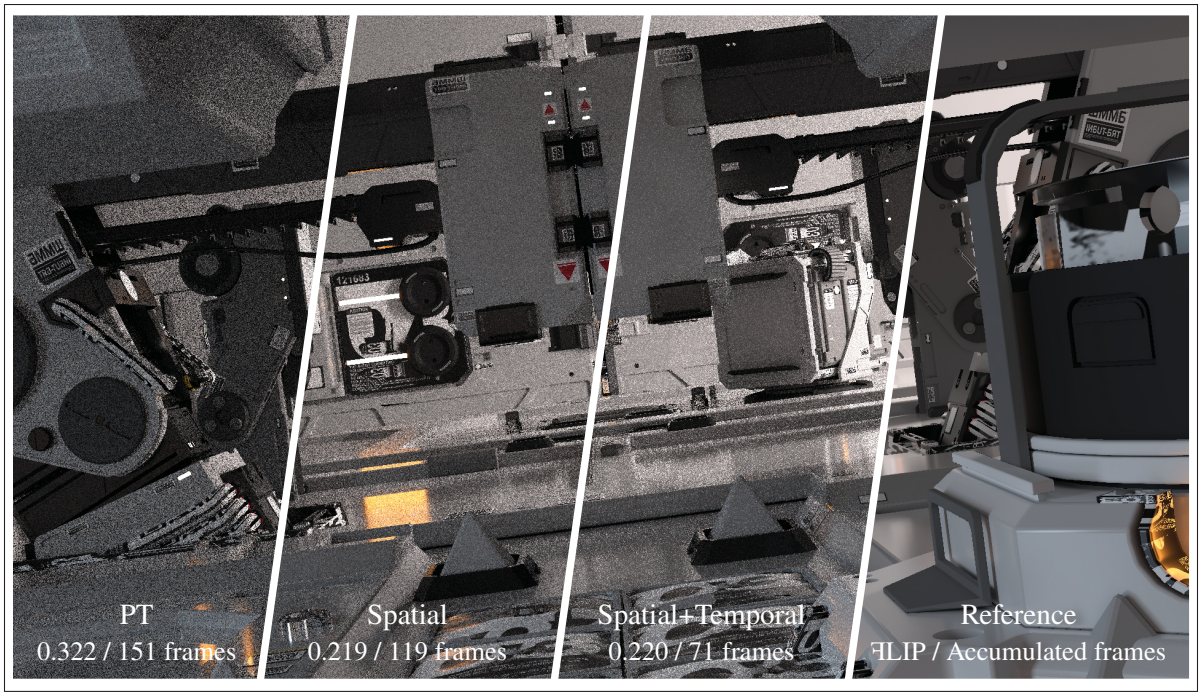


Figure 2.2 Equal-time (5 s) comparison between path tracing, spatial reuse, and spatial+temporal reuse on the ZERO DAY scene

Algorithm 2.4 demonstrates the ReSTIR process: given a reservoir pool which can be sourced either from spatial or temporal pixels, we first shift each sample into the canonical pixel's domain. We then evaluate the generalized RIS weights, using these to perform reservoir merging in the correct domain. This gives us the final sample chosen by RIS, whose UCW we then evaluate and store. The resulting reservoir can then be used directly in an MC estimator, or fed to more iterations of ReSTIR for further refinement.

Figure 2.2 shows the efficiency gains possible with ReSTIR. Even though each frame takes longer to run, over an equal period of time ReSTIR provides significant variance reduction benefits compared to path tracing. We measure the difference with respect to the reference image with HDR-FLIP (Andersson, Nilsson, Shirley & Akenine-Möller, 2021), which we choose as it closely approximates how error is visually perceived by humans. Note that although the spatiotemporal reuse has slightly higher metrics owing to temporal correlation artifacts, there is a visually appreciable reduction in noise on the grey box in the top-centre of the image.

2.3.1 Confidence weighting

Ideally, we would also like to control the importance of ReSTIR samples through their MIS weights, giving lower weight to samples that contain less information and heavily weighting those with more information. Because one RIS sample represents M samples in aggregate, it is effectively as powerful as M samples, a concept which we will call the RIS sample's *effective sample count* (Wyman *et al.*, 2023). This motivates *confidence weights*, which favours samples that have seen repeated reuse and hence have higher effective sample counts. If each ReSTIR candidate X_i has effective sample count M_i, then we can assign confidence weights $\gamma_i = M_i$ and reformulate the generalized balance heuristic from Equation (2.3) as

$$m_i(y) = \frac{\gamma_i \hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} \gamma_j \hat{p}_{\leftarrow j}(y)}. \tag{2.4}$$

New samples with no history of reuse are assigned confidence weight 1. We then assign the confidence weight $\gamma_s = \sum_i \gamma_i$ to the returned ReSTIR sample, allowing for slow accumulation of confidence proportional to the effectiveness of the sample. However, if the stored confidence is too large, then we will end up with a highly correlated result (Lin *et al.*, 2022). Hence, in practice we cap γ_s with a maximum value γ_{cap}; a larger value favours variance reduction, while a lower value favours minimizing correlation artifacts. Algorithm 2.5 implements ReSTIR with the inclusion of confidence weights.

2.3.2 Advanced MIS weights

Although the balance heuristic introduced in Equation (2.3) minimizes variance and is robust across a wide variety of target functions, its $O(M^2)$ cost is discouraging. Evaluation of the shift mapping within the MIS weight also requires visibility testing, further increasing its cost. Advanced MIS weight formulations have thus been proposed that keep the efficiency of the balance heuristic while minimizing its cost. We cover two such formulations, contribution MIS weights and pairwise MIS weights.

Algorithm 2.5 ReSTIR with confidence weights

```
struct Reservoir
    Y ← ∅
    W ← 0
    γ ← 0                                          // Stored confidence weight
    w_sum ← 0
    function update(x_i, w_i, γ_i)
        w_sum ← w_sum + w_i
        γ ← γ + γ_i                                // Accumulate confidence weight
        if Random() < w_i / w_sum then
            Y ← x_i
        end if
    end
end
function Resample(r_1, r_2, ··· , r_M)
    Reservoir r
    for i ← 1 to M do
        Y_i ← T_{c←i}(r_i.Y)
        w_i ← GeneralizedRISWeight(c, i, r_i)
        r.update(Y_i, w_i, r_i.γ)
    end for
    if r.Y ≠ ∅ then
        r.W ← (1 / p̂(r.Y)) r.w_sum
    end if
    r.γ ← min(r.γ, γ_cap)                          // Cap output confidence
    return r
end
```

Contribution MIS weights. Bitterli *et al.* (2020) introduced contribution MIS weights in the RIS context, which allows for only evaluating an advanced MIS weight once for the selected sample. Let s be the selected index and

$$Z(y) = \{1 \le i \le M \mid \hat{p}(y) > 0\}. \tag{2.5}$$

We then define the following UCW for $Y = T_{c\leftarrow s}(X_s)$, according to Lin *et al.* (2022):

$$W_Y = \left(\frac{\gamma_s(Y)}{m_s(Y)}\right)\frac{1}{\hat{p}(Y)}\sum_{i=1}^{M} w_i, \tag{2.6}$$

where $\gamma_i(x)$ is a user-defined function that satisfies $\sum_{i\in Z(y), y\in T_{c\leftarrow i}(\operatorname{supp} X_i)} \gamma_i(y) = 1$. We can then choose $m_i(y) = 1/M$ and $\gamma_i(y) = 1/|Z(y)|$ so that evaluating the w_i is cheap and only the computation of $\gamma_s(Y)$ in W_Y is left, which results in a total computational complexity of $O(M)$ for the RIS estimator. Alternatively, we can let $\gamma_i(y)$ be the generalized balance heuristic, which gives better results when many candidate distributions are involved (Bitterli *et al.*, 2020).

Pairwise MIS weights. Pairwise MIS weights were introduced in Bitterli (2022), taking advantage of the fact that in rendering there is usually one candidate distribution that produces much better samples than the others. This is usually the distribution associated with the canonical pixel c, which we refer to as the *canonical distribution*; it is then required that $\operatorname{supp}(p_c) \supseteq \operatorname{supp}(f)$, guaranteeing that the resulting estimator will be unbiased. The generalized pairwise MIS weights are given as:

$$\begin{aligned} m_i(y) &= \frac{\gamma_i \hat{p}_{\leftarrow i}(y)}{\left(\sum_{k\neq c}^{M}\gamma_k\right)\hat{p}_{\leftarrow i}(y) + \gamma_c \hat{p}_c(y)}, && (i \neq c) \\ m_c(y) &= \sum_{j\neq c}^{M}\left(\frac{\gamma_j}{\sum_{k\neq c}^{M}\gamma_k}\right)\frac{\gamma_c \hat{p}_c(y)}{\left(\sum_{k\neq c}^{M}\gamma_k\right)\hat{p}_{\leftarrow j}(y) + \gamma_c \hat{p}_c(y)}. \end{aligned} \tag{2.7}$$

This can result in bias under certain situations where $\hat{p}_c(y)$ is relatively small, which in turn underweighs the canonical sample. To rectify this, a defensive form is also provided:

$$\begin{aligned} m_i(y) &= \left(\frac{\sum_{k\neq c}^{M}\gamma_k}{\sum_{k=1}^{M}\gamma_k}\right)\frac{\gamma_i \hat{p}_{\leftarrow i}(y)}{\left(\sum_{k\neq c}^{M}\gamma_k\right)\hat{p}_{\leftarrow i}(y) + \gamma_c \hat{p}_c(y)}, && (i \neq c) \\ m_c(y) &= \frac{\gamma_c}{\sum_{k=1}^{M}\gamma_k} + \sum_{j\neq c}^{M}\left(\frac{\gamma_j}{\sum_{k=1}^{M}\gamma_k}\right)\frac{\gamma_c \hat{p}_c(y)}{\left(\sum_{k\neq c}^{M}\gamma_k\right)\hat{p}_{\leftarrow j}(y) + \gamma_c \hat{p}_c(y)}. \end{aligned} \tag{2.8}$$

2.3.3 Advanced shift mappings

Although the reconnection shift mapping from Section 1.5 is simple to implement, it is problematic when naively applied to practical path tracing. For example, on a near-specular surface reconnection will likely produce a direction that falls on the tail of the BSDF, which causes the reused sample to have a large UCW, resulting in fireflies. Reconnection can also produce a direction that falls completely outside of the specular lobe, causing the shift mapping to fail entirely. Figure 2.3 presents several shift mapping strategies that seek to avoid these pitfalls, which we detail below.

Random sequence replay. An alternative way to view shift mappings is through the lens of sample generation, in that the sample in the mapped domain is generated with a one-to-one correspondence from the sample in the original domain. One such shift mapping, random sequence replay, keeps track of the tangent-space random sequence used to generate vertices in the base path $\bar{\mathbf{x}}$, and reuses that sequence to generate vertices for the offset path $\bar{\mathbf{y}}$ with respect to the vertex normal (Kelemen, Szirmay-Kalos, Antal & Csonka, 2002). It works well for glossy BSDFs where reconnection is likely to fail, ensuring the offset path will be generated similarly to the base path. However, this correlation is lost if the offset path samples from a BSDF that differs from the equivalent vertex in the base path (Hua *et al.*, 2019). If random numbers $u_{\mathbf{x}}$ and $u_{\mathbf{y}}$ are used to generate $\omega_i^{\mathbf{x}}$ and $\omega_i^{\mathbf{y}}$ respectively, then the Jacobian for random sequence replay is given as

$$\left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}u_{\mathbf{y}}}\right|\left|\frac{\mathrm{d}u_{\mathbf{y}}}{\mathrm{d}u_{\mathbf{x}}}\right|\left|\frac{\mathrm{d}u_{\mathbf{x}}}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \frac{p(\omega_i^{\mathbf{x}})}{p(\omega_i^{\mathbf{y}})}. \tag{2.9}$$

Half-vector copy. Half-vector copy computes the half-vector between path segments in the base path $\bar{\mathbf{x}}$, and preserves this half-vector to sample the next path vertex in the offset path $\bar{\mathbf{y}}$ (Kettunen *et al.*, 2015). The half-vector for a vertex $\mathbf{x}_i$ is calculated as

$$\mathbf{h}_{\mathbf{x}_i} = \frac{\omega_o + \omega_i}{\|\omega_o + \omega_i\|}. \tag{2.10}$$

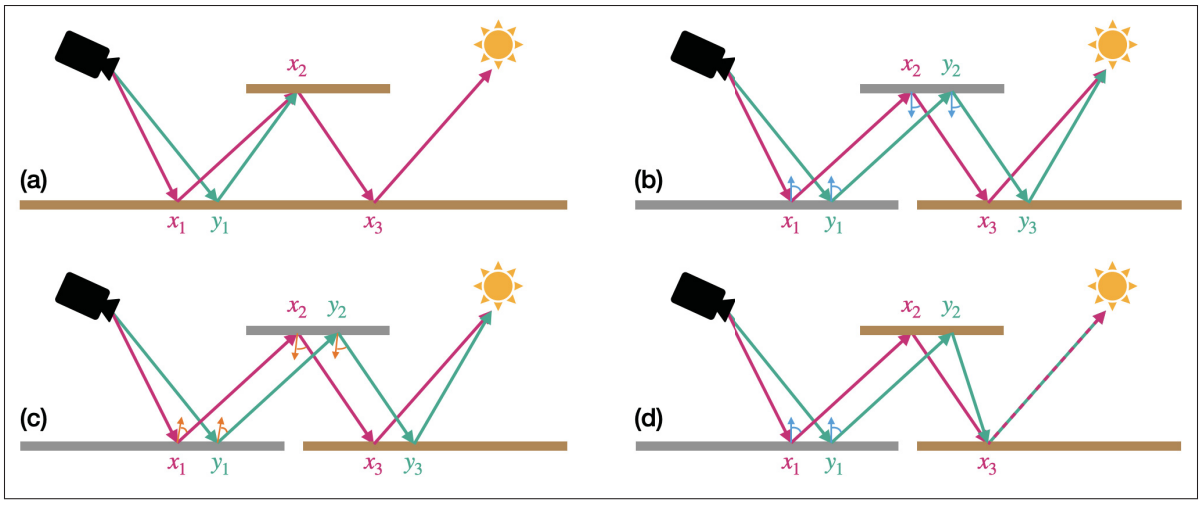


Figure 2.3 Various shift mappings used in ReSTIR. We showcase **(a)** reconnection shift, **(b)** random sequence replay, **(c)** half-vector copy, and **(d)** the hybrid random sequence replay/reconnection mapping introduced in Hua *et al.* (2019)

Because it is independent of the BSDF for any given surface hit, it does not suffer from the same correlation issues as random sequence replay. The Jacobian for half-vector copy depends on whether the incident direction ω_i is reflected or refracted; in the former case, we have

$$\left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \frac{\omega_o^{\mathbf{y}} \cdot \mathbf{h}_{\mathbf{y}_i}}{\omega_o^{\mathbf{x}} \cdot \mathbf{h}_{\mathbf{x}_i}}, \tag{2.11}$$

and for the latter case, we have

$$\left|\frac{\mathrm{d}\omega_i^{\mathbf{y}}}{\mathrm{d}\omega_i^{\mathbf{x}}}\right| = \frac{\left|\omega_i^{\mathbf{y}} + \frac{\mathbf{n}_{\mathbf{y}_{i+1}}}{\mathbf{n}_{\mathbf{y}_i}}\omega_o^{\mathbf{y}}\right|}{\left|\omega_i^{\mathbf{x}} + \frac{\mathbf{n}_{\mathbf{x}_{i+1}}}{\mathbf{n}_{\mathbf{x}_i}}\omega_o^{\mathbf{x}}\right|}\frac{\omega_i^{\mathbf{x}} \cdot \mathbf{h}_{\mathbf{x}_i}}{\omega_i^{\mathbf{y}} \cdot \mathbf{h}_{\mathbf{y}_i}}. \tag{2.12}$$

Hybrid shift mappings. Kettunen *et al.* (2015) introduce a hybrid approach that uses half-vector copy up until the soonest diffuse base path vertex that is also preceded by a diffuse vertex, at which point reconnection shift is used. This allows for the efficient reuse promised by reconnection shift while avoiding its incompatibility with specular BSDFs, for which half-vector copy is a better candidate. Hua *et al.* (2019) replace the half-vector copy in Kettunen *et al.*'s shift

mapping with random sequence replay, which is faster and avoids having to store half-vectors at every single offset vertex, saving considerable amounts of memory. Like in Equation (1.36), the Jacobian of the offset path is simply the product of the Jacobians at the individual offset vertices for both hybrid shift mappings, allowing us to easily reuse the existing formulas above.

2.4 ReSTIR variations

The basic ReSTIR method described above can be applied to a variety of lighting estimators. We cover its specific applications to the direct illumination, global illumination, and path tracing forms of the rendering equation.

2.4.1 ReSTIR DI

In the most basic form of ReSTIR, introduced in Bitterli *et al.* (2020), only direct illumination is considered. Taking the rendering equation from Equation (1.4) and limiting it to paths of length 2, we derive the direct lighting equation:

$$L_{direct}(\mathbf{x}_0 \leftarrow \mathbf{x}_1) = \int_A f(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) L_e(\mathbf{x}_1 \leftarrow \mathbf{x}_2) \, \mathrm{d}A(\mathbf{x}_2) \, . \tag{2.13}$$

In this case, our target function $\hat{p}(\bar{\mathbf{x}})$ is the integrand

$$\hat{p}(\bar{\mathbf{x}}) = f(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) L_e(\mathbf{x}_1 \leftarrow \mathbf{x}_2). \tag{2.14}$$

The only free vertex in $\bar{\mathbf{x}}$ is $\mathbf{x}_2$, the variable of integration; this will also be the vertex used for the reconnection shift mapping. In other words, given $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2]$ and $\bar{\mathbf{y}} = [\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2]$, we would like to generate the offset path $[\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_2]$.

ReSTIR DI starts with the construction of the canonical sample pool. The original per-pixel path samples are usually generated with NEE sampling, and we save them in a screen-space reservoir buffer. Following the architecture of Wyman & Panteleev (2021), temporal reuse is conducted first. The reservoir from the current pixel is merged with that of the motion-matched pixel from

the previous frame using the respective frames' reservoir buffers, saving the output reservoir in the current frame's buffer. Spatial reuse is then conducted by sampling a small number of neighbouring pixels within a given reuse radius, and merging the current pixel's reservoir with the reservoirs of the sampled pixels, all of which are loaded from the current frame's reservoir buffer. Both temporal and spatial reuse use reconnection shift with the current pixel's path as the base path, and ReSTIR DI does not require complex shift mapping machinery as path sampling is performed in the area domain, in which case the Jacobian of the shift mapping is 1.

2.4.2 ReSTIR GI

ReSTIR GI (Ouyang *et al.*, 2021) extends the base ReSTIR algorithm to path tracing by expanding the reservoir to store the indirect illumination contribution and the surface normals of the primary and secondary hits. This requires the use of BSDF sampling to generate the paths, so all probability calculations are performed in the solid-angle domain, which in turn necessitates the introduction of the Jacobian. Like ReSTIR DI, ReSTIR GI depends on the reconnection shift mapping, albeit to reconnect the secondary hits instead of light vertices.

Spatial reuse is most effective when neighbouring pixels share similar target functions, but this is difficult to achieve with the global illumination due to greater path divergence. Boissé (2021) and Zhang & Wang (2023) hence propose storing reservoirs in a spatial hash grid, so that reservoirs that are spatially coherent in screen-space are also spatially coherent in world-space.

2.4.3 ReSTIR PT

Although ReSTIR GI allows for indirect illumination sample reuse, its reliance on the reconnection shift mapping leaves it susceptible to fireflies and failed shifts from glossy surfaces, which present themselves as boiling artifacts. ReSTIR PT (Lin *et al.*, 2022) replaces the reconnection shift with the hybrid shift proposed in Hua *et al.* (2019) to reduce these artifacts. This shift mapping also includes distance and roughness conditions which reject reconnections that are too short or reuse on BSDF lobes that are too specular, avoiding correlation from geometric and

material singularities and guaranteeing a good candidate sample pool. Despite its robustness, ReSTIR PT's hybrid shift mapping is quite expensive and difficult to implement.

Previously, one disadvantage of ReSTIR PT (as well as with ReSTIR GI) was that the reservoirs could only store indirect illumination. This meant that in order to reuse direct illumination, one had to run both ReSTIR DI and ReSTIR PT passes, heavily impacting performance. Furthermore, with a 1920×1080 pixel render target over 900MB of video memory would have to be allocated just for the reservoir buffers alone,[1] adding to the compute costs. Lin, Kettunen & Wyman (2026) show that it is possible to merge the direct and indirect illumination reservoirs, leading to significant memory savings and performance improvements from only needing to run a single ReSTIR pass.

2.4.4 Other improvements

Many further improvements on the base ReSTIR algorithm have been investigated in recent years. We describe some of them below.

Complex light transport. Lin, Wyman & Yuksel (2021) apply ReSTIR to participating media by introducing reuse for scattering paths and a novel temporal reprojection algorithm. ReSTIR FG (Kern, Brüll & Grosch, 2024) enhances ReSTIR with a final gathering pass taken from photon mapping, while ReSTIR BDPT (Hedstrom, Kettunen, Lin, Wyman & Li, 2025) demonstrates the application of ReSTIR to bidirectional path tracing; both techniques greatly improve ReSTIR's performance in caustic rendering.

Reducing correlation artifacts. Correlation in ReSTIR often arises from specular surfaces behaving poorly with the reconnection shift. Conditional ReSTIR (Kettunen, Lin, Ramamoorthi, Bashford-Rogers & Wyman, 2023) reduces correlation by reusing over subpaths which can be chosen to avoid including the reconnection vertex. Alternatively, Sawhney *et al.* (2023) propose perturbing the samples in the temporal reuse chain with Markov Chain Monte Carlo, essentially

[1] Combined, the reservoirs require roughly 450MB, but a backbuffer must be kept to allow for temporal reuse, doubling the overall memory footprint.

adding a slight amount of variance to reduce correlation bias. Zeng *et al.* (2025) also merge ReSTIR and path guiding to improve the initial sample quality provided to the former and the quality of the guiding distributions for the latter, further reducing correlation artifacts.

Improvements to temporal reuse. One of the primary challenges of temporal reuse is matching primary hits between frames, since the motion-mapped current frame pixel does not necessarily lie neatly within past frames' pixel grids. Area ReSTIR (Zhang, Lin, Kettunen, Yuksel & Wyman, 2024) searches within a 2×2 pixel grid for the best match to the backprojected pixel using RIS, while Liu, Lin, Kettunen, Wyman & Ramamoorthi (2025) propose forward-projecting the past frame's primary hit into the current frame's pixel. Both methods improve the correlation between the current and past frame's primary hits and hence the chances of successful reuse.

Improvements to spatial reuse. Meanwhile, the primary challenge facing spatial reuse is the success of choosing a contributing reservoir, especially in areas of poor lighting or heavy geometric occlusion where such reservoirs are sparse. One might think to naively sample reservoirs based on their contents, but this results in bias (Wyman *et al.*, 2023). Instead, various techniques for improving the chances of selecting a contributing reservoir have been proposed: Salaün *et al.* (2025) suggest importance sampling a histogram constructed over the spatial reuse region, Hedstrom, Kettunen, Lin, Wyman & Li (2026) present an unbiased method for importance sampling surrounding reservoirs with stochastic MIS weights, and Junkins, Kettunen, Lin, Ramamoorthi & Wyman (2026) introduce a geometric compatibility score for neighbour selection. These methods serve as concurrent work to our own. Spatial reuse can also be made more efficient with reciprocal neighbour selection (Lin *et al.*, 2026), which halves the number of shift mappings that must be computed and thus reduces runtime.

2.5 Discussion

ReSTIR is an extremely good variance reduction technique; by most metrics, it offers up to a five-fold improvement. However, it suffers from several structural issues that lead to undesirable

visual artifacts. Chief among them is a phenomenon known as boiling, where spatial reuse causes the contribution of fireflies to be shared across pixels in the reuse radius, resulting in circular spots that are noticeably brighter than surrounding areas; we provide a detailed explanation in Appendix B. These fireflies arise from low-probability events that in turn have a very large UCW; usually, these events stem from specular materials which have near-delta BSDFs. Because of their narrow lobes, shifting to or from vertices with these materials often fails, so RIS is unable to importance sample the integrand.

Another source of issues is geometric singularities within the reuse radius. If the vertices used in the reconnection shift have substantially different normals or positions, then the shift mapping is more likely to fail, leading to increased bias. Bitterli *et al.* (2020) suggests using heuristics to reject reuse in these cases, but this leads to structural noise artifacts located around these geometric discrepancies. It would be better if $\mathbf{x}'$ is chosen to guarantee geometric similarity to $\mathbf{x}$, leading to more instances of successful reuse.

Finally, if the scene contains lighting that is difficult to sample, there will be a low density of contributing reservoirs. This makes it difficult for spatial reuse to perform efficiently, since RIS depends on selecting high-quality reservoirs to effectively importance sample the target function. Furthermore, the few samples that do end up being successful tend to overcontribute to the final estimator, resulting in highly visible correlation artifacts (Sawhney *et al.*, 2023).

We attempt to tackle the firefly issue by regularizing the reconnection shift to reduce its failure rate. We do so by modelling the reconnection vertex as a virtual spherical Gaussian light (Tokuyoshi, 2015), mollifying the narrow BSDF lobe and increasing the likelihood of a successful shift mapping; we describe our method and present results in Appendix A. We approach the latter two issues by building upon mipmaps (Williams, 1983) through applying them to the reservoir buffer, resulting in a novel method of spatially clustering reservoirs which we detail in Chapter 3.

CHAPTER 3

MIPMAPPED RESERVOIRS

3.1 Introduction

Generalized resampled importance sampling selects one sample from a stream of samples, and transforms it to be representative of the population (Lin *et al.*, 2022). In traditional ReSTIR, the population is the set of reservoirs corresponding to all previous frames in the case of temporal reuse, and the set of reservoirs within a given radius in the case of spatial reuse. It is prohibitively expensive to fully consider the latter population, so in practice a small subset of reservoirs is sampled to be considered for reuse. However, whatever sampling strategy is chosen must be wary of low-quality reservoirs, which will always exist no matter the initial path generation technique, and will increase variance if sampled. A non-uniform sampling strategy would mitigate these issues, but attempting to importance sample the surrounding reservoirs will introduce bias (Wyman *et al.*, 2023). This leads to a classic bias-variance tradeoff, which has historically been decided in favour of increasing variance to decrease bias.

One alternative approach is to simply ensure that every possible reservoir is high-quality. Ideally, we would like to ensure that outlier and empty contributions are amortized over their surrounding reservoirs. We can group neighbouring reservoirs into spatial clusters, which we then make as large as possible in order to increase the likelihood of having at least one nonempty reservoir per cluster. However, larger clusters also mean higher correlation, since there are fewer reservoirs to choose from for a given spatial reuse radius. Therefore, it would be ideal to simultaneously construct clusters of multiple sizes to allow us to adaptively choose between them. Mipmapping gives us an algorithm to construct these clusters; by recursively splitting the source image into 2×2 pixel clusters per level, we can merge reservoirs over and over again with geometrically increasing cluster sizes.

In this chapter, we review the fundamentals of mipmapping and its precursor, texture filtering. We then present a novel algorithm for mipmapping reservoirs, and explain two practical strategies

to implement it. We also present a non-recursive adaptive clustering algorithm with an improved response to geometric and material discontinuities. Finally, we compare these algorithms against ReSTIR GI (Ouyang *et al.*, 2021) and analyze the results.

3.2 Background

Similar to how Zhang, Lin, Wyman & Yuksel (2025) perform ReSTIR over shadow maps, we can view the entire screen-space reservoir buffer as a texture map where the value of each texel is equal to that reservoir's contribution. This motivates us to treat undesirable artifacts such as fireflies and empty reservoirs as high-frequency data, which in turn suggests the application of signal processing techniques to filter these artifacts out of the final result.

3.2.1 Texture filtering

The usage of texture maps in modern rendering frameworks allows for detailed visualizations of surfaces and materials. Unfortunately, naive implementation of texture maps leads to highly distracting aliasing artifacts as when the underlying continuous signal is discretized, high-frequency details end up being undersampled (Heckbert, 1986). One solution, arising from signal processing, is to remove the offending frequencies from the original signal with a low-pass function, leading to the process of texture filtering. Given that 2D texture maps are rasterized over texels as discrete samples $\mathbf{t}_{u,v}$, they cannot be filtered as-is. We thus take the definition of the texture function $t(u, v)$ from Heckbert (1989), which gives us a formulation over $\mathbb{R}^2$:

$$t(u, v) = \sum_{u'} \sum_{v'} \delta(u - u')\delta(v - v')\mathbf{t}_{u',v'}, \tag{3.1}$$

where $\delta(0) = \infty$ and $\delta(x) = 0, x \neq 0$ is the Dirac delta function. Note that $t(\cdot)$ is non-zero if and only if its input is a texel location, i.e. it is given non-negative integral inputs within the bounds of the texture map. We can then view texture filtering as a convolution between t and the

filtering operator f giving a continuous texture function:

$$t_f(u,v) = (t \otimes f)(u,v) = \iint_{\mathbb{R}^2} t(u',v')\, f(u-u', v-v')\ \mathrm{d}u'\mathrm{d}v'. \tag{3.2}$$

Expanding Equation (3.2) over the Dirac delta functions gives the much more practical formulation

$$t_f(u,v) = \sum_{u'} \sum_{v'} f(u-u', v-v')\mathbf{t}_{u',v'}. \tag{3.3}$$

Traditionally, the filtering operator is separated into a reconstruction filter representing coordinate interpolation and an analysis filter representing a frequency-limiting function (Nehab & Hoppe, 2014). However, we choose to combine both into f for simplicity.

3.2.2 Stochastic texture filtering

Stochastic texture filtering (STF) uses Monte Carlo integration over the filtering operator, thus bypassing the need to read every texel in the filter domain (Pharr *et al.*, 2024). It has its roots in stochastic volume filtering, which is used in volume rendering applications due to the additional dimensionality introduced with that problem (Hofmann, Hasselgren, Clarberg & Munkberg, 2021). STF can be performed before or after shading, which respectively refer to whether the input textures are filtered before being passed to the BSDF or whether the output radiance is filtered post-integration. In the filter-after-shade context, we use this form of the rendering equation for pixel j based on Equation (1.15):

$$I_j = \left(\int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; t(u_{\mathbf{x}_1}, v_{\mathbf{x}_1})\big) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k \right) \otimes f, \tag{3.4}$$

where the throughput function is parameterized on the texture function $t(u_{\mathbf{x}_1}, v_{\mathbf{x}_1})$ of the primary hit, which is used for the shading term $f(\mathbf{x}_0 \leftarrow \mathbf{x}_1 \leftarrow \mathbf{x}_2)$. Expanding this convolution gives the STF estimator

$$\langle I_j \rangle_{STF} = \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; t(u'_{\mathbf{x}_1}, v'_{\mathbf{x}_1})\big) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k\,, \tag{3.5}$$

assuming (u'_j, v'_j) are sampled proportionally to f. Alternatively, the texture function can be convolved directly to produce the rendering equation for filter-before-shade:

$$I_j = \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; (t \otimes f)(u_{\mathbf{x}_1}, v_{\mathbf{x}_1})\big) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k \,. \tag{3.6}$$

STF can also be applied to reuse texel values as demonstrated in Wronski, Pharr & Akenine-Möller (2025), producing a consistent albeit biased estimator.

3.2.3 Mipmapping

Mipmapping is a technique that recursively constructs filtered, downsampled images from a source image map (Williams, 1983). Historically, it has been used with texture maps to avoid aliasing artifacts; however, it has also been adapted to filter normal maps, height maps, and numerical BRDFs (Toksvig, 2005; Tevs, Ihrke & Seidel, 2008; Gauthier *et al.*, 2022). Each constructed image is known as a *level-n mip*, with the source image being level 0, and each texel of the level-$(n+1)$ mip constructed according to the following texture filtering formula:

$$t^{n+1}_{\text{mip}}(u, v) = \Big(t^n_{\text{mip}} \otimes f_{\text{mip}}\Big)(u, v), \tag{3.7}$$

where t^n_{mip} is the texture function for the level-n mip, and f_{mip} serves as both a reconstruction and aliasing filter (Blinn, 1989). Practical choices of f_{mip} usually have a finite domain $\mathcal{D}$, which means that the level-$(n+1)$ mip merges all texels within $\mathcal{D}$ of the level-n mip into one, creating an output image map with reduced dimensions. Usually, mipmapping is performed with a box filter over a 2×2 texel grid; evaluating Equation (3.7) in this case gives:

$$\text{mip}^{n+1}_{u,v} = \frac{1}{4} \sum_{u' \in \{0,1\}} \sum_{v' \in \{0,1\}} \text{mip}^n_{2u+u', 2v+v'} \,. \tag{3.8}$$

Under this construction, each subsequent mip level halves the dimensions of the previous mip level, making mipmaps extremely memory efficient. A full set of mipmaps only takes $1 + \frac{1}{4} + \frac{1}{16} + \cdots = \frac{4}{3}$ the memory of the original source image, and since mip levels can be



Figure 3.1 Example of a mipmap, with the original image at left. Note the lack of aliasing artifacts in each progressively downsampled level

loaded individually, practical memory usage is almost always lower. Figure 3.1 demonstrates the memory efficiency of mipmaps and their application towards preventing aliasing.

Sampling mipmaps is more involved than sampling the original texture, since one must convert the input texture coordinates into both the appropriate mipmap level and texel (Heckbert, 1986). The simple nearest-neighbour approach chooses the closest mip level and mip texel, but produces clear mip level artifacts. Bilinear filtering improves on this by interpolating the input coordinates over a 2×2 texel grid on the chosen mipmap, and trilinear filtering further interpolates over mipmap levels. These three methods are well-supported in hardware, allowing for mipmap sampling to be performed with little extra cost.

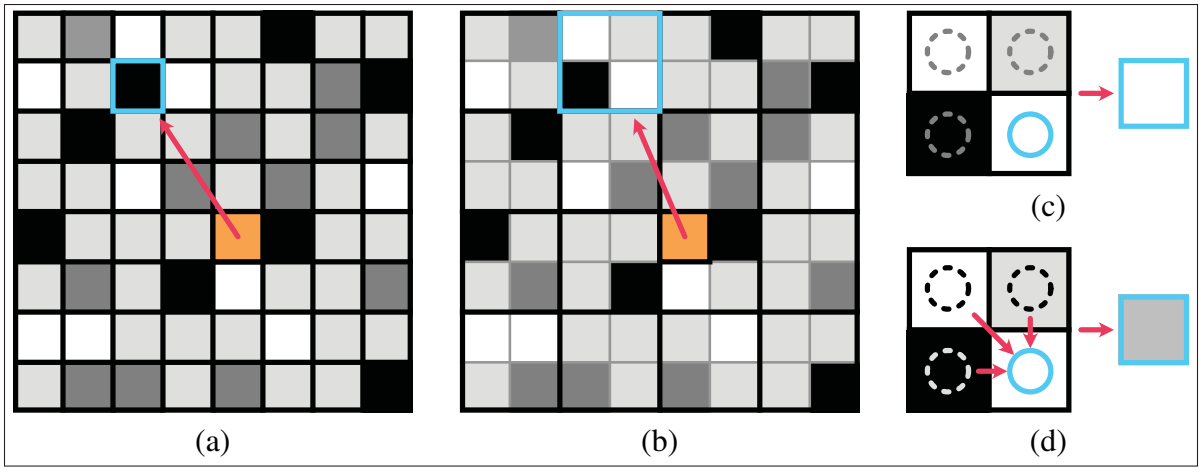


Figure 3.2 **(a)** ReSTIR has no control over the spatial samples that it selects, which can lead to attempts to reuse low-quality samples. **(b)** By grouping reservoirs into 2×2 clusters à la mipmapping, the likelihood of choosing a low-quality sample is lessened. The target density and weights of each cluster can be calculated through **(c)** reservoir selection, where one of the four reservoirs is chosen to represent the entire cluster, or **(d)** reservoir merging, which produces a single representative reservoir

3.3 Reservoir mipmapping

One of the main challenges towards improving spatial reuse in ReSTIR is the potential for selecting low-quality samples. We can exploit the observation that samples contained in spatially close reservoirs usually have similar contributions and geometry, especially if stored in world-space, and that shift mappings between them are more likely to succeed (Hua *et al.*, 2019). This inspires us to extend the mipmapping algorithm to operate on reservoirs, generating a single high-quality reservoir for each mipmap cluster that reflects the source reservoirs' overall contribution, which we visualize in Figure 3.2. Applying clustering algorithms to ReSTIR is not new; Hong *et al.* (2025) improve caustics rendering by tiling the screen-space reservoir buffer to spatially bound manifold exploration (Jakob & Marschner, 2012). Our overall approach also bears similarities to Area ReSTIR (Zhang *et al.*, 2024), but whereas they limit themselves to improving temporal reuse with the framebuffer-scale reservoir buffer, we operate over the spatial domain and on multiple mip levels.

Algorithm 3.1 Reservoir mipmapping

```
   Input: A screen-size buffer of Reservoir and the desired maximum mipmap level n.
   Output: An array of mipmapped reservoir buffers.
 1 function MipmapReservoirs(sourceReservoirs, n)
 2     reservoirs ← Array of pointer to Reservoir of size n
 3     for i ← 1 to n do
           // Create downscaled reservoir for level-i mip
 4         prevReservoirs ← if i = 1 then sourceReservoirs else reservoirs[i − 1]
 5         dims ← ⌊prevReservoirs.dims/2⌋
 6         reservoirs[i] ← new Reservoir[dims]
 7         for u ← 1 to dims.width do
 8             for v ← 1 to dims.height do
                   // Filter reservoirs with the mipmapping operator
 9                 D ← {(2u, 2v), (2u, 2v + 1), (2u + 1, 2v), (2u + 1, 2v + 1)}
10                 r ← f_mip({u − u′, v − v′ | (u′, v′) ∈ D}; sourceReservoirs)
11                 reservoirs[i][(u, v)] ← r
12             end for
13         end for
14     end for
15     return reservoirs
16 end
```

Algorithm 3.1 details the process of constructing the mipmapped reservoir buffer, which we call *reservoir mipmapping*. Taking from Equation (3.7), we use t^n_{mip} to index into each mipmap level's buffer, where $n = 0$ gives the initial framebuffer-sized reservoir buffer. We can then apply texture filtering theory to the reservoir buffer, allowing us to reason through the reservoir mipmapping process as we would with texture mipmapping. However, because we cannot average over reservoirs, we must include the necessary merging and/or sampling procedures within f_{mip} itself. f_{mip} also returns a reservoir corresponding to the output texel, rather than a colour contribution as would be associated with texture data.

Since our approach stands before final shading, it is appropriate to view it through the lens of filter-before-shade, where the filtering algorithm is the combination of mipmapping and spatial reuse. Recursively substituting Equation (3.7) into the filter-before-shade equation from

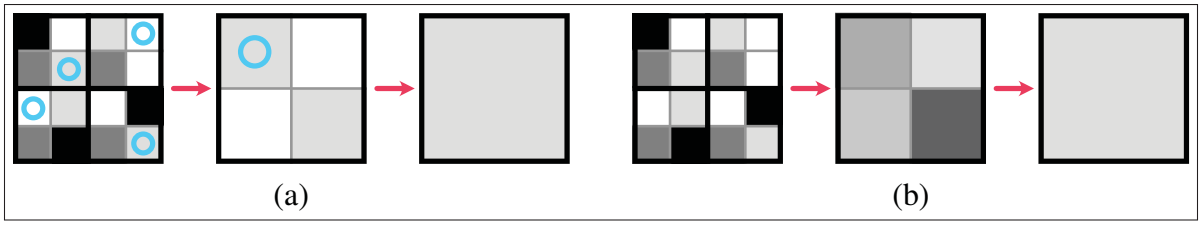


Figure 3.3 **(a)** Reservoir selection and **(b)** reservoir merging performed over multiple mipmap levels

Equation (3.6) gives:

$$\begin{aligned} I_j &= \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; t^n_{\text{mip}}(u_j, v_j)\big) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k \\ &= \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; (t^{n-1}_{\text{mip}} \otimes f_{\text{mip}})(u_j, v_j)\big) L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k \\ &\vdots \\ &= \int_{\mathcal{P}} W_e^j(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) T\big(\bar{\mathbf{x}}_k; (t^0_{\text{mip}} \otimes f_{\text{mip}} \otimes^{n-1} f_{\text{mip}})(u_j, v_j)\big) \\ &\qquad L_e(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)\, \mathrm{d}\bar{\mathbf{x}}_k \,. \end{aligned} \tag{3.9}$$

MC integration of Equation (3.9) is unbiased so long as the estimator for $(t^0_{\text{mip}} \otimes f_{\text{mip}} \otimes^{n-1} f_{\text{mip}})(u_j, v_j)$ itself is unbiased, i.e. $\langle t^n_{\text{mip}} \rangle$ is an unbiased estimator for t^{n-1}_{mip} for all $n \geq 1$; proof of this is provided in Appendix C. Since t^0_{mip} returns the original reservoirs, which themselves store unbiased estimates of the rendering equation, $t^n_{\text{mip}} = t^{n-1}_{\text{mip}} \otimes f_{\text{mip}}$ can be viewed as a form of filter-after-shade, allowing us to calculate the next mipmap level via MC estimation of f_{mip}. We can then use the machinery of stochastic texture filtering from Section 3.2.2 to produce an unbiased estimator.

As the mipped reservoirs no longer share the same dimensions as the screen, we must find a way to index them that maps screen pixels (x, y) onto mip texels (u, v). Assuming a 2×2 texel grid filter domain, we derive the following mapping to and from level-n mips:

$$(u, v) = (\lfloor x/2^n \rfloor, \lfloor y/2^n \rfloor), \qquad (x, y) = (2^n u, 2^n v). \tag{3.10}$$

Within the ReSTIR pipeline, we insert reservoir mipmapping between temporal reuse and spatial reuse. The spatial reuse pass may then query the mipmapped reservoir buffer as if it were still sized to the framebuffer, mitigating the need for further modifications. This is akin to the nearest-neighbour approach for sampling texture mipmaps. The reduced size of the mipmapped reservoir buffer also allows the spatial reuse pass to load and query it more efficiently, potentially bringing further uplifts in performance.

The choice of f_{mip} is highly influential towards determining the outcome of reservoir mipmapping. We present reservoir merging and reservoir selection as options, both of which are robust methods that produce acceptable results, and of which we give a brief visual overview in Figure 3.3. We follow discussion of these two strategies with an improved technique for choosing the mipmap domain.

3.3.1 Reservoir merging

One way to sample the filtering operator is through weighted reservoir sampling, which Pharr *et al.* (2024) call filter reservoir sampling. This permits unbiased MC estimation of discrete filters, or complex filters that have intractable sampling pdfs. We extend filter reservoir sampling by applying it directly to the reservoirs themselves with RIS, and by introducing shift mappings to account for the change in domain between the source texels. Lin *et al.* (2022) show that RIS can be chained infinitely many times to produce an unbiased estimator, thus guaranteeing that f_{mip} is unbiased per Equation (3.9). Our process is detailed in Algorithm 3.2.

Mipmapped reservoir merging is similar to executing successive layers of spatial reuse as detailed in Bitterli *et al.* (2020), but with dimension reduction at each step. Furthermore, reservoir merging produces a buffer of reservoirs *for* spatial reuse with the goal of improving the input quality of the latter. However, care must be taken to avoid performance deficiencies from the merging process. In particular, attention must be paid towards the choice of canonical reservoir and the choice of MIS weights. Regardless, reservoir merging inherits RIS's benefits, particularly its robust construction and ability to be chained.

Algorithm 3.2 Mipmapped reservoir merging

```
function MipmapReservoirMerge(r_1, r_2, r_3, r_4)
    Reservoir r_out
    c ← MipmapChooseCanonical()
    for i ← 1 to 4 do
        w_i ← m_i(r_i.Y) p̂_{c←i}(r_i.Y) r_i.W
        r_out.update(r_i.Y, w_i)
    end for
    r_out.W ← r_out.w_sum / p̂(r_out.Y)
    return r_out
end
```

Choosing the canonical reservoir. Because f_{mip} requires us to shift between the multiple texels that it covers, we must choose a canonical reservoir to evaluate these shifts against. The simplest approach for doing so is to take a fixed texel from the 2×2 source grid, e.g. the top-left texel. We also repeat this for each subsequent mip level, so that the canonical texel takes the form $(2^n u, 2^n v)$ for each mip level $n \geq 1$.

Algorithm 3.3 Strategies for choosing the canonical reservoir

```
function MipmapChooseCanonicalFixed()
    // Top-left pixel
    return 1
end
function MipmapChooseCanonicalUniform()
    return a random integer in [1, 4]
end
```

Fixing the canonical texel guarantees screen-space temporal stability, and can easily be adjusted for camera movement via backprojection or forward splatting (Zhang *et al.*, 2024; Liu *et al.*, 2025). However, a fixed canonical texel means that the canonical reservoir will stay constant across all mip levels, which we can readily verify with Equation (3.10). This quickly leads to aliasing artifacts, since the sampling frequency of the canonical reservoir is aligned with the input frequency of f_{mip}. We can mitigate these artifacts by selecting our canonical reservoir with

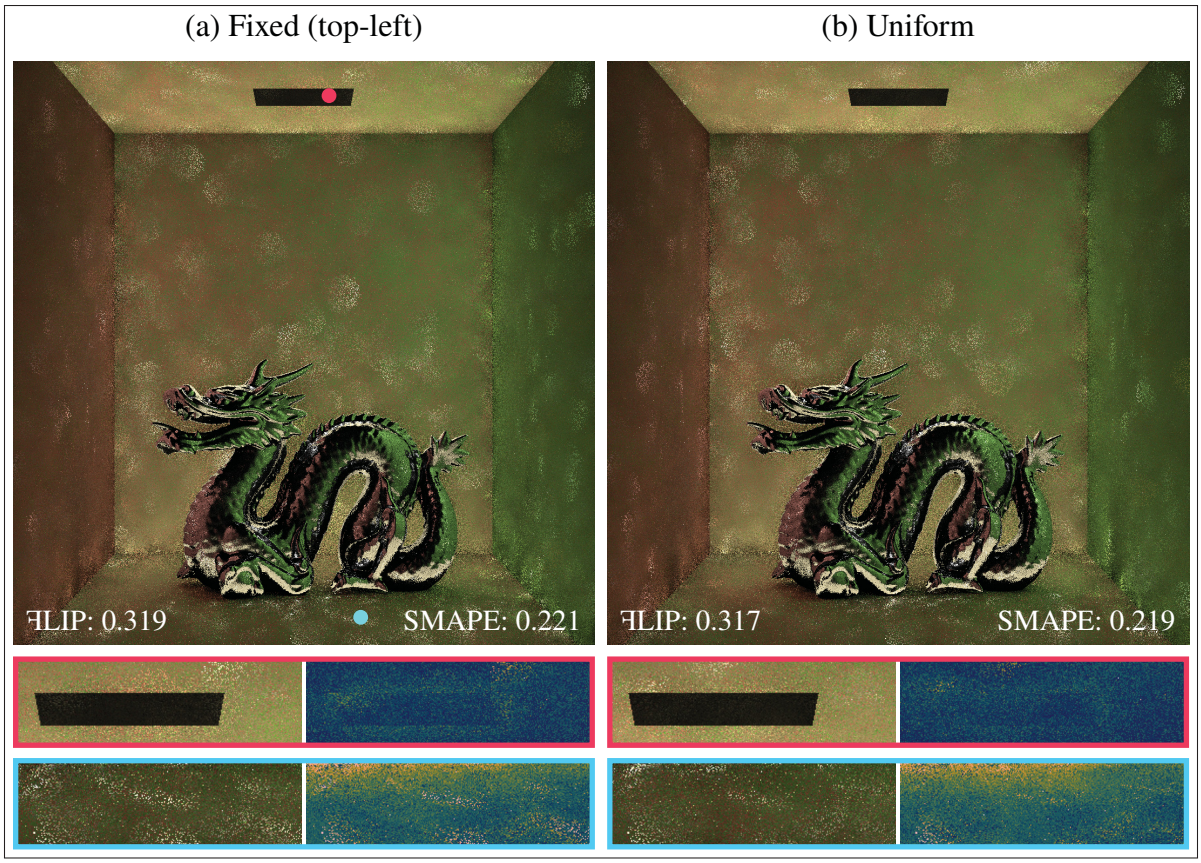


Figure 3.4 **(a)** Fixing the location of the canonical sample results in highly visible correlation artifacts propagating down the mip chain, worsening the quality of the final image. **(b)** Uniformly choosing the canonical samples reduces the maximum intensity of these artifacts. The bottom rows depict the SMAPE metric in false colour. Scene is a closed Cornell Box supplemented with the Stanford Dragon

uniform sampling, which we implement alongside fixed sampling in Algorithm 3.3. Uniform sampling does come at the cost of slightly increased variance, which we demonstrate in Figure 3.4 with false colour differences taken with the symmetric mean absolute percentage error (SMAPE) metric. We choose SMAPE as it visualizes the direct error distribution better than ꟻLIP, and define it as

$$\text{SMAPE} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \frac{\left|\langle I_j \rangle - I_j\right|}{\left|\langle I_j \rangle\right| + \left|I_j\right|}, \tag{3.11}$$

where $\mathcal{J}$ is the domain of screen-space pixels, I is the true value of the rendering equation (1.15) represented by the reference image, and $\langle I \rangle$ is its estimate represented by the test image.

MIS weights. A full evaluation of the mip chain for a level-n mip requires $O(n^2)$ queries in the balance heuristic, since each successive mip level halves the dimensions of the reservoir buffer. As the balance heuristic itself requires $O(n^2)$ queries, our method does not incur any further asymptotic costs. We can improve upon this by using pairwise MIS weights from Equation (2.8), which gives a total cost of $O(n)$ queries. We thus prefer the latter method for evaluating the m_i.

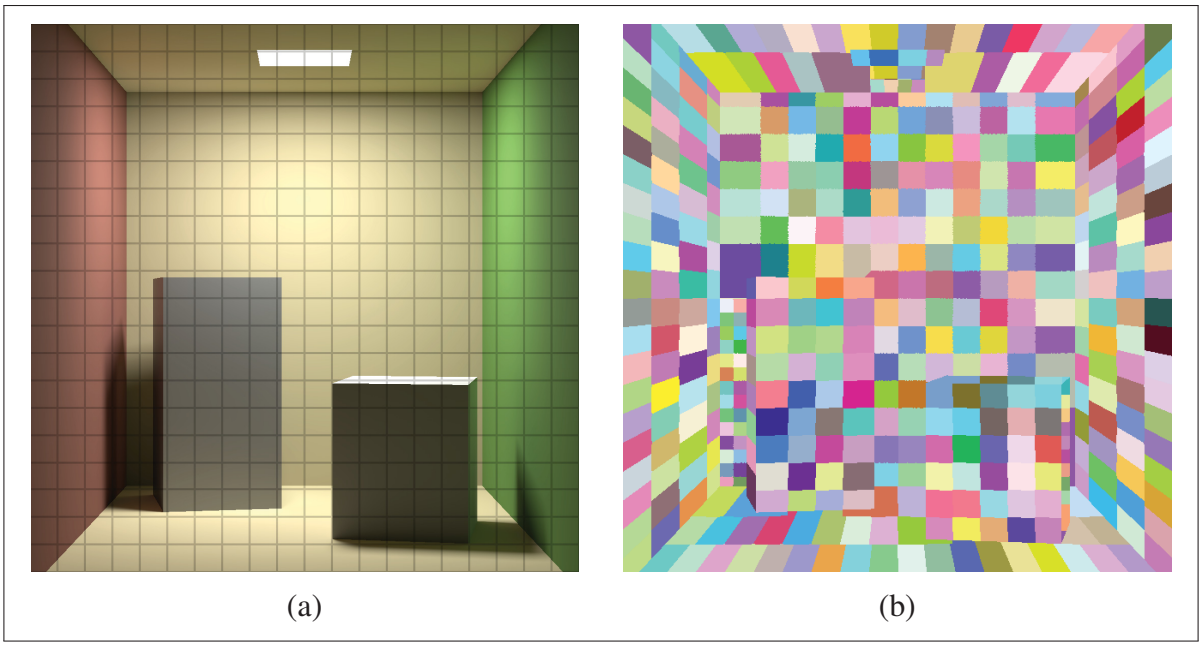
3.3.2 Reservoir selection

Despite providing an unbiased result, reservoir merging is computationally heavy due to the need to evaluate the shift mapping multiple times. Therefore, as a cheaper alternative we propose merely importance sampling the reservoirs in the filter domain, similar to the technique discussed in Wronski *et al.* (2025).

Algorithm 3.4 Mipmapped reservoir selection

```
1  function MipmapReservoirSelect(r_1, r_2, r_3, r_4)
2      s ← 0
3      w_sum ← 0
4      for i ← 1 to 4 do
5          w_i ← (1/4) p̂(r_i.Y) r_i.W
6          w_sum ← w_sum + w_i
7          u ← Random()
8          if u < w_i / w_sum then
9              s ← i
10         end if
11     end for
12     r_s.W ← w_sum / p̂(r_s.Y)
13     return r_s
14 end
```

A simple way to sample clusters is to sample one single representative reservoir from within each cluster, which avoids the complexity of having to calculate shift mappings. Given the filter domain $\mathcal{D} = \{q_1, q_2, q_3, q_4\}$ with corresponding screen-space reservoirs r_1, r_2, r_3, r_4, we choose one of the constituent pixels $q_s \in \mathcal{D}$, modify the UCW of its reservoir r_s according to



(a) (b)

Figure 3.5 **(a)** In a screen-space mipmap grid, a texel can contain geometric discontinuities when it covers multiple objects or primitives. **(b)** Adaptive clustering adjusts for the scene geometry, and can also be made aware of surface orientation

RIS, and return r_s. Choosing q_s can then be done via weighted reservoir sampling using naive MIS weights. This guarantees that each output reservoir will have non-zero contribution as long as there exists at least one non-empty source reservoir. We demonstrate our technique in Algorithm 3.4.

3.3.3 Adaptive clustering

The aforementioned mipmapping methods operate on a simple 2×2 screen-space grid. While suitable for hierarchical generation of mips, the lack of geometry and material awareness means that each cluster can cover multiple geometric objects or materials in the scene, as Figure 3.5a demonstrates. These discontinuities in turn cause the shift mappings to fail, resulting in increased variance and artifacts. One approach would be to discard problematic clusters as ReSTIR does with dissimilar pixels (Bitterli *et al.*, 2020); however, owing to the eventual large screen-space

Algorithm 3.5 SLIC-based reservoir selection

```
1  function SLICReservoirSelect(sourceReservoirs, SLICClusters)
2      reservoirs ← Reservoir[SLICClusters.dims]
3      for u ← 1 to reservoirs.cols do
4          for v ← 1 to reservoirs.rows do
5              D ← SLICClusters[(u, v)]
6              r ← f_mip({u − u′, v − v′ | (u′, v′) ∈ D}; sourceReservoirs)
7              reservoir[(u, v)] ← r
8          end for
9      end for
10     return reservoirs
11 end
12 function MipmapReservoirsSLIC(sourceReservoirs)
13     Generate SLIC clusters
14     reservoirs ← SLICReservoirSelect(sourceReservoirs, SLIC clusters)
15     return reservoirs
16 end
```

coverage of mipmap texels and the fine surface geometry often seen in real-world scenes, this technique is impractical.

Instead of using a simple fixed domain, we can apply context-aware adaptive clustering methods. For example, Lu *et al.* (2024) cluster pixels with geometrically-similar primary hits using a geometry-aware version of the SLIC superpixel algorithm (Achanta *et al.*, 2012), the effects of which can be seen in Figure 3.5b. We further modify Lu *et al.*'s approach, expanding their metric to account for differences in pixel-space and world-space positions as well as the materials of the primary hit, and using one single iteration of gSLICr (Ren, Prisacariu & Reid, 2015) on the GPU for increased performance. We follow this by importance sampling the reservoirs contained within each cluster to produce our output reservoir similar to Section 3.3.2, which produces similar results while avoiding the geometric discontinuities; this constitutes our mipmap filtering function. Algorithm 3.5 demonstrates this approach, and we note its implementation details below.

GPU-based cluster importance sampling. Although gSLICr provides a performant method for clustering on the GPU, it does not provide for importance sampling a single reservoir from the cluster. We therefore devise an algorithm for importance sampling reservoirs from SLIC clusters; we first prefix sum the RIS weights of each reservoir in the cluster, then use parallel importance sampling on these weights to select one reservoir. To our knowledge, this algorithm is novel and has not been introduced elsewhere. An implementation in the Slang shading language (He, 2021) is provided in Appendix D.

Choosing the initial cluster size. SLIC accepts various choices for the initial cluster width, which is used as guidance to define a square boundary within which the algorithm will check for matching pixels. In our implementation, we limit this to either 8, 16, or 32 pixels. Although values outside this range are theoretically possible, lower values are too small for the clusters to properly adapt and higher values do not provide any visual benefit due to the magnitude of the involved correlation artifacts. These sizes are equivalent to level-3, level-4, and level-5 mips respectively, allowing us to compare the two techniques.

Components of the distance metric. Our distance metric is composed of three distinct components — screen-space distance, geometric distance (a combination of world-space distance and normal vector similarity), and a boolean distance based on the materials of the compared reservoirs' primary hits:

$$\begin{gathered}\text{dist}_{ss} = \left\| j_{\mathbf{x}} - j_{\mathbf{y}} \right\|^2, \quad \text{dist}_g = \|\mathbf{x} - \mathbf{y}\|^2 + \left|\mathbf{n}_{\mathbf{x}} \cdot \mathbf{n}_{\mathbf{y}}\right|^{-1}, \\ \text{dist}_m = \begin{cases} 1 & \text{if } \mathbf{x} \text{ and } \mathbf{y} \text{ have the same material,} \\ 0 & \text{otherwise.} \end{cases}\end{gathered} \tag{3.12}$$

The complete distance metric is then given as

$$\text{dist}_{SLIC} = \text{dist}_m \sqrt{w_g \text{dist}_g + w_{ss} \text{dist}_{ss}}, \tag{3.13}$$

where w_g and w_{ss} are weights that are given to the SLIC algorithm as hyperparameters. By default, we set both weights to 1, though they may be carefully optimized per-scene.

Table 3.1 Frame times for different variants of our algorithms, compared against base path tracing and ReSTIR GI

Scene	PT	ReSTIR GI	Mip1	Mip3	Sel3	SLIC8
Living Room	30.34 ms	40.22 ms	42.81 ms	43.11 ms	42.10 ms	41.92 ms
Bistro	36.48 ms	52.93 ms	56.51 ms	55.70 ms	53.41 ms	53.31 ms
Staircase	17.31 ms	26.65 ms	28.56 ms	28.28 ms	27.31 ms	27.17 ms
Veach Ajar	19.68 ms	30.13 ms	29.44 ms	29.70 ms	29.17 ms	29.02 ms
Zero Day	25.25 ms	39.15 ms	40.51 ms	40.72 ms	39.31 ms	39.19 ms

3.4 Results

We implement our methods on the Falcor real-time rendering framework (Kallweit *et al.*, 2022), and generate our results using a RTX 4070 GPU and a Ryzen 5 3600 CPU. Importantly, we implement mipmapped reservoirs on ReSTIR GI (Ouyang *et al.*, 2021) rather than ReSTIR PT (Lin *et al.*, 2022). This allows us to emphasize the benefits and drawbacks of our approach with regards to handling fireflies, a problem that the latter can only mitigate at great computational cost. We also choose to present our results and comparisons without temporal reuse, to emphasize the improvements in spatial reuse that we achieve.

We include results for level-1 and level-3 mipmapped reservoir merging, level-3 mipmapped reservoir selection, and adaptive clustering with 8px wide initial clusters, which we respectively label as Mip1, Mip3, Sel3, and SLIC8. We do not include results for level-1 mipmapped reservoir selection, as they are closely proxied by those for level-1 mipmapped reservoir merging. Our results are rendered with path length 5 (i.e. 3 indirect bounces) to a 1920×1080 pixel framebuffer. On top of the Cornell Box with Dragon scene from Figure 3.4, we use the following additional test scenes: Living Room, Staircase, Veach Ajar from Bitterli (2016); Bistro from Amazon Lumberyard (2017); and Zero Day from Pharr (2023).

Our implementation of ReSTIR GI is similar to the unified direct lighting and global illumination technique introduced in Lin *et al.* (2026), in that we are able to reuse between paths of varying

Table 3.2 Frame breakdown of each technique when rendering the Bistro scene

Pass	ReSTIR GI	Mip1	Mip3	Sel3	SLIC8
GBuffer			0.69 ms		
GenerateSamples			34.56 ms		
MipmapReservoirs					
Mipmap(1)	-	2.85 ms	2.85 ms	1.21 ms	-
Mipmap(2)	-	-	0.83 ms	0.34 ms	-
Mipmap(3)	-	-	0.26 ms	0.1 ms	-
MipmapReservoirsSLIC					
GenerateClusters	-	-	-	-	0.35 ms
SLICReservoirSelect	-	-	-	-	1.18 ms
SpatialResampling	16.45 ms	17.18 ms	15.28 ms	15.28 ms	15.3 ms
FinalShading			1.23 ms		
Δ_t with ReSTIR GI	-	**3.58 ms**	**2.77 ms**	**0.48 ms**	**0.38 ms**

lengths, including direct lighting paths. However, since we do not consider direct lighting in our global illumination paths, our implementation is less efficient.

General comparisons. Figure 3.6 compares Mip3, Sel3, and SLIC8 against baseline ReSTIR GI over a multitude of scenes. All three methods show significant improvement over baseline ReSTIR GI, with reservoir merging outperforming reservoir selection at the same mip level at the cost of some bubbling. Adaptive clustering with SLIC superpixels also performs similarly to reservoir selection of the equivalent mipmap level across all scenes.

We provide frame timings in Table 3.1; of the three proposed methods, SLIC clustering is the fastest, followed by reservoir selection and reservoir merging. We also provide a single-frame breakdown of per-pass timings with total time deltas against ReSTIR GI for the Bistro scene in Table 3.2. Notably, although the mipmapping process adds several milliseconds to the frame time, this is offset by improved performance in the spatial resampling pass, resulting in reservoir selection and SLIC clustering having sub-millisecond total time deltas.

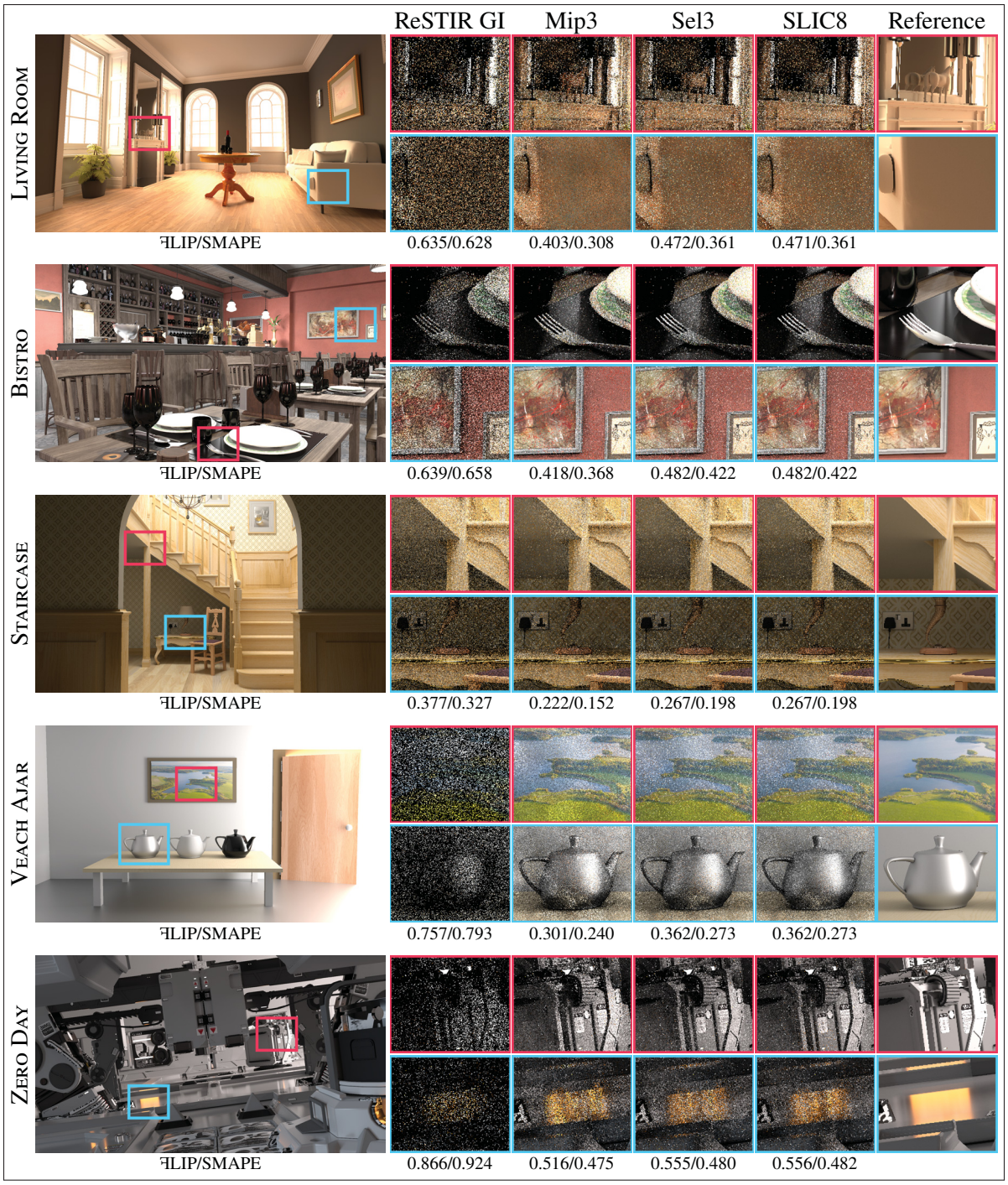


Figure 3.6 One-sample renders over various scenes, comparing ReSTIR GI with level-3 equivalents of reservoir merging, reservoir selection, and adaptive clustering

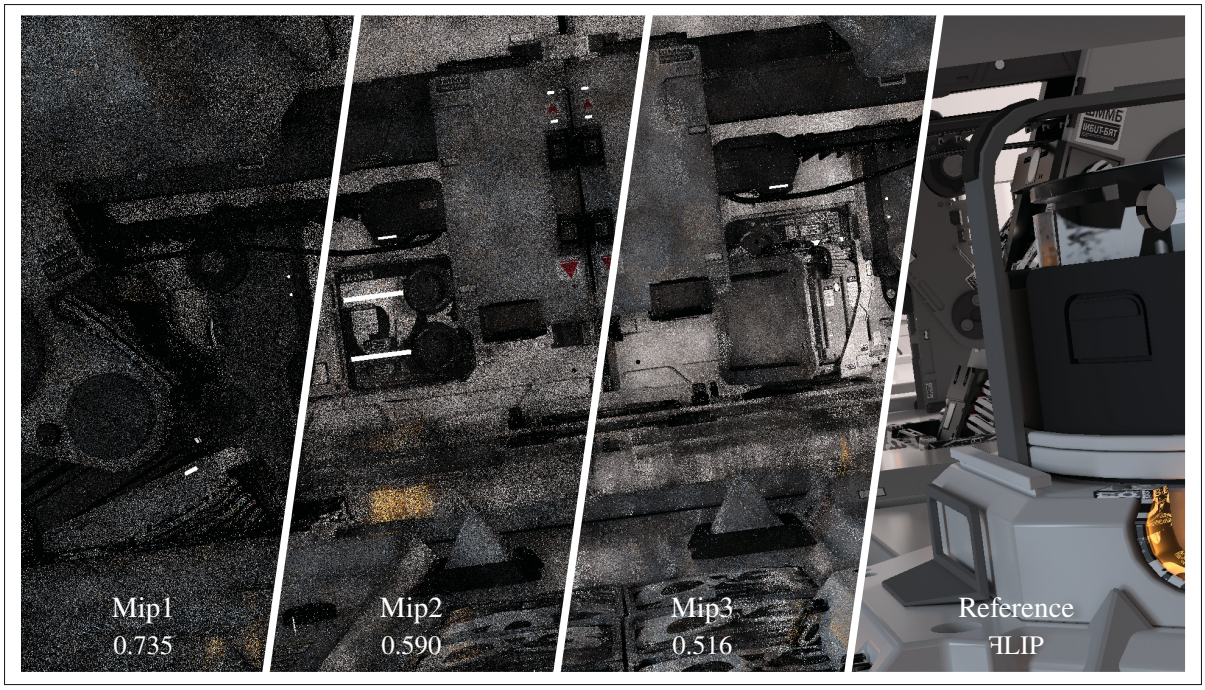


Figure 3.7 Comparison of different mip levels on the ZERO DAY scene. As mip level increases, variance decreases, but bubbling is more strongly visible

Choosing the mip level. Although increasing the mipmap level improves variance metrics on diffuse surfaces, it leads to bubbling artifacts from fireflies present in the initial reservoirs. The variance reduction is most strongly felt when the initial reservoir buffer has a high density of empty reservoirs, usually corresponding to difficult paths. For example, VEACH AJAR and ZERO DAY rely heavily on indirect illumination from occluded emitters for the majority of each primary hit's lighting contribution. Figure 3.7 demonstrates the effects of increasing mip level on reservoir merging using the latter scene.

Performance on specular surfaces and caustics. When the primary hit falls on a specular surface where shift mappings are more likely to fail, reservoir merging results in correlation artifacts that outweigh the variance reduction. These artifacts can be mitigated with reservoir selection, albeit with a mild increase in variance. On the other hand, caustics see moderate improvement from higher mipmap levels, which can be explained by the mipmapping process being naturally suited for the inherent spatial correlation of caustics (Xu, Wang & Wang, 2023).

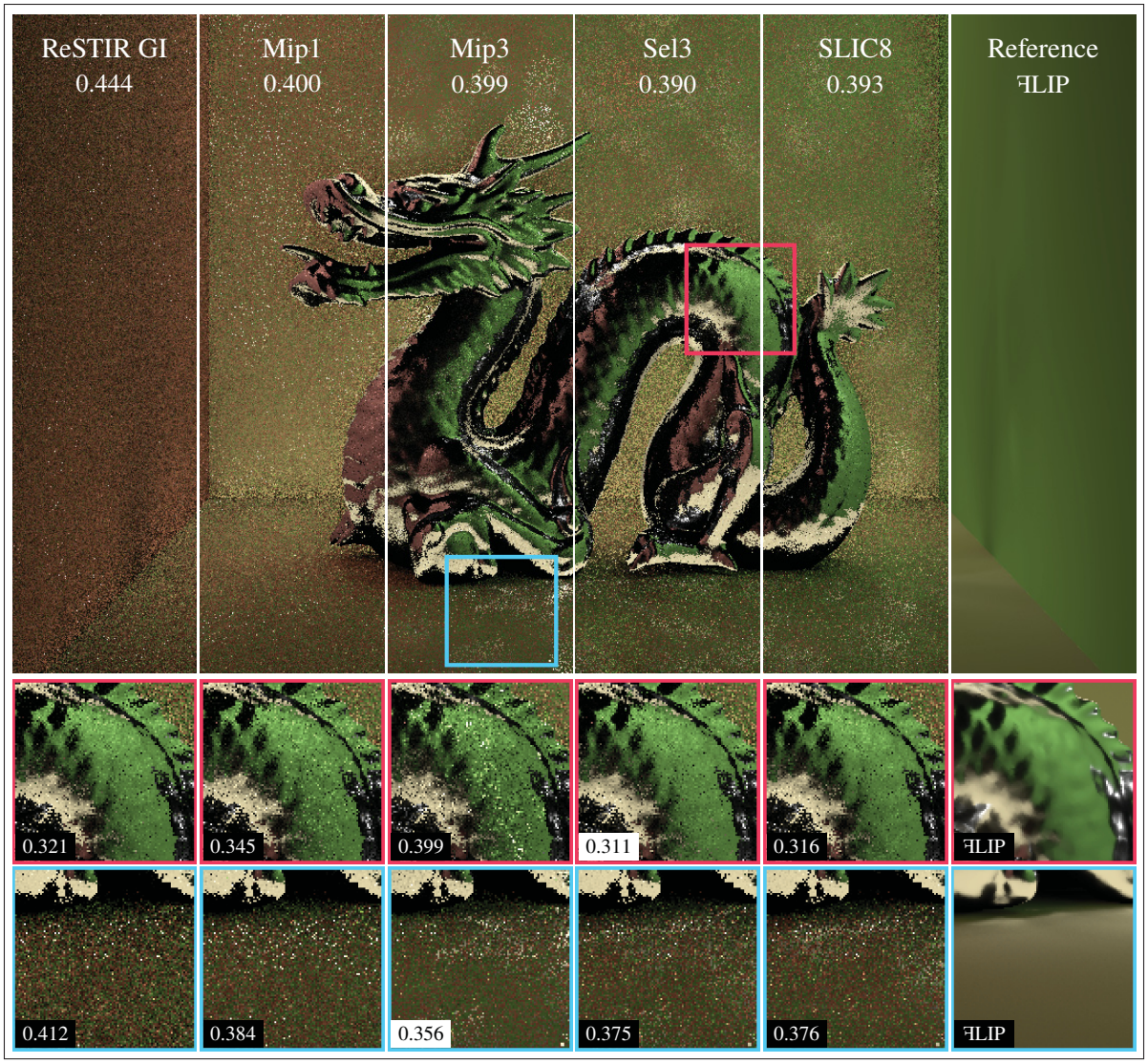


Figure 3.8 Breakdown of results for specular surfaces and caustics. Path length 3, indirect lighting, 1 sample. Lowest ꟻLIP score for each crop indicated with inverted colours. Note that selection performs the best on specular surfaces, while mipmapping performs worse especially as the mipmap level increases; however, when rendering caustics, the latter method delivers the best results

Figure 3.8 demonstrates this dichotomy, comparing level-1 and level-3 reservoir merging, level-3 reservoir selection, and adaptive clustering with 8px wide initial clusters against base ReSTIR GI.

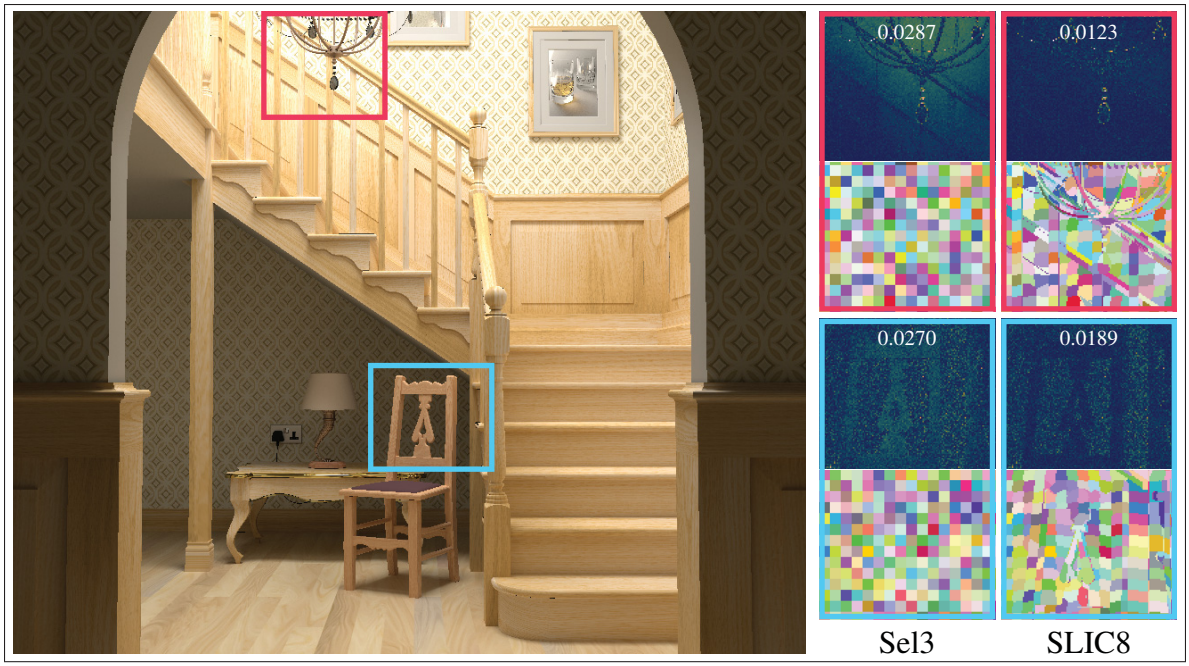


Figure 3.9 Equal-time (10 s) rendering of the STAIRCASE scene comparing level-3 reservoir selection versus SLIC8, with weights $w_g = 1$ and $w_{ss} = 0.001$ for the latter. All error values shown in SMAPE. False-colour values scaled 5× for contrast

Optimizing adaptive clustering. The SLIC weight components dist_g and dist_{ss} are sensitive to the scale of the scene, which makes global, scene-independent values for w_g and w_{ss} inappropriate for production use-cases. In particular, we find that adjusting the weights of the components of the distance metric to favour geometric distance by increasing w_g and/or decreasing w_{ss} in Equation (3.13) improves results in regions of severe geometric discontinuities, as demonstrated in Figure 3.9. However, these improvements are usually negated by slightly higher variance elsewhere, such that the overall change is only visible after being accumulated for relatively long time periods. Furthermore, if the regions containing these discontinuities are small enough to fit within a single cluster, there will be no added benefit to weight adjustment. Therefore, one must consider the geometric properties of the target scene and the magnitude of any variance reduction benefits when choosing how to optimize adaptive clustering.

Consistency and bias. Figure 3.10 shows equal-time convergence graphs over all six test scenes for Mip1, Mip3, Sel3, and SLIC8. We compare these methods with spatial reuse-only

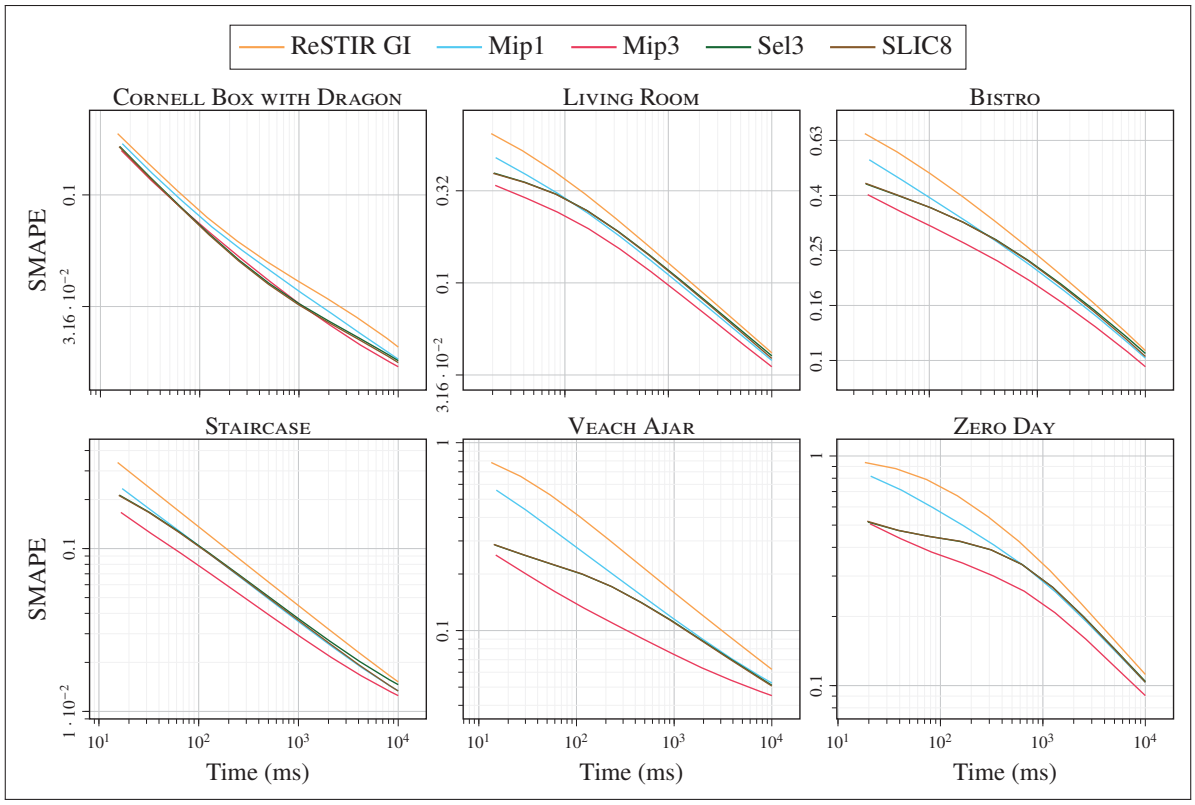


Figure 3.10 Equal-time convergence plots for spatial reuse over our test scenes, measuring SMAPE versus render time. We compare against ReSTIR GI without temporal reuse as our baseline

ReSTIR GI accumulated over the same time frame; the removal of temporal reuse allows for accurate comparison, since our methods operate independently of it in the spatial domain. Despite our methods being consistent estimators, they are not necessarily unbiased; Figure 3.11 demonstrates that reservoir selection (including with adaptive clustering) produces a biased estimator, especially with more complex scenes where many-light sampling is required. Per-pixel bias is specifically calculated as

$$\text{bias}_j = \frac{1}{|C|} \sum_{c \in C} \langle I_{j,c} \rangle - I_{j,c}, \tag{3.14}$$



Figure 3.11 Bias shown by Sel3 on Living Room and Bistro respectively after accumulating for 10 seconds. Red regions denote negative bias, green regions positive bias, and yellow regions mixed bias between RGB channels

where C is the set of colour channels in the image, and the definitions of I and $\langle I \rangle$ are as in Equation (3.11). This bias can be explained by the importance sampling process relying on the contents of the source reservoirs, which leads to a biased output reservoir (Wyman *et al.*, 2023).

CONCLUSION AND RECOMMENDATIONS

The quality of ReSTIR's output is directly dependent on the quality of its source samples and its shift mappings. When faced with a high density of zero-contribution samples, ReSTIR often fails catastrophically, providing minimal improvement over basic path tracing. Our novel mipmapped shift mapping framework avoids this by filtering the reservoir buffer, thus increasing the quality of samples available to downstream processes. In turn, we demonstrate two practical shift mappings within this framework, and complete our contribution with a novel GPU importance sampling algorithm that permits their implementation over adaptive clusters. Our contributions provide significant variance reduction improvements at little additional cost.

4.1 Future works

Our application of stochastic texture filtering to reservoir mipmapping produces some interesting lines of exploration. For one, it appears that it is possible to treat the screen-space reservoir buffer much like any ad-hoc colour texture. This suggests that all the advancements in texture filtering over the past 50 years may be applicable to reservoirs, allowing us to further increase sample quality. Furthermore, it may be possible to view ReSTIR as a continuous texture filtering operator in $\mathbb{R}^2$; Zhang *et al.* (2024) and Liu *et al.* (2025) provide precedence for this. Finally, differentiable rendering (Li, 2019) requires a rendering algorithm such as path tracing to be run over multiple iterations, resulting in a total runtime that tends to be practically infeasible; Chang *et al.* (2023) accelerate the differentiable rendering process with ReSTIR, and applying our method may yield further performance improvements.

Our method can very much be improved through further exploration. The temporal stability of fixed versus uniform canonical samples in Section 3.3.1 has yet to be tested, and it may prove the case that the former is more suited for temporal reuse than the latter. The gSLICr parallel importance sampling algorithm in Section 3.3.3 has also only been applied to reservoir selection; a future work may be able to extend it to reservoir merging, allowing for more flexible reservoir mipmapping domains. Finally, it may be possible to solve the bias of reservoir selection by calculating proxies for the RIS weights, thus removing any correlation between them and their

respective reservoirs' contents. Alternatively, it may be the case that its use of naive MIS weights leads to bias, in which case a suitable replacement must be found.

Hedstrom *et al.* (2026) provide an improved method for selecting reservoirs in screen-space which also accounts for the quality of the source samples. Notably, their method closely parallels our proposed reservoir selection method, albeit in an unbiased manner and without reducing the dimensions of the reservoir buffer. It may be possible to apply their method within the framework of stochastic texture filtering to avoid having to sample every cell in a given cluster, merging the cost benefits of their stochastic selection with the quality benefits of our mipmapping algorithm.

Adaptive clustering performance is dependent on the chosen values of the hyperparameters, since the evaluated distance metric is sensitive to the scale of the scene. Per-scene hyperparameter tuning would thus allow for improved clustering and better avoidance of geometric and material singularities. However, since the rendering equation does not depend on the clustering hyperparameters, automated tuning would have to rely on gradient-free optimization algorithms such as CMA-ES (Hansen, 2023); applying such methods is an area of interest for increasing the feasibility of adaptive clustering. The limited benefit seen from adaptive clustering in the tested scenes may also be because they are not well suited for clustering with SLIC, and improvements could be more visible with a different clustering algorithm, such as the spatial hash maps used in Alber *et al.* (2025) and Zhang & Wang (2023).

APPENDIX A

RESERVOIR REGULARIZATION WITH VIRTUAL SPHERICAL GAUSSIAN LIGHTS

Reconnection shift mappings are more likely to fail on glossy BSDFs due to the narrowness of their specular lobes, resulting in fireflies. One approach is to regularize such BSDFs, but this causes the output image to be biased. However, since path reuse is independent of the final shading step, we propose that it is possible to regularize the shift mapping without biasing the final output. We thus implement a method for regularizing the reconnection shift mapping by augmenting reservoir reuse with virtual spherical Gaussian lights, which shows some promise in reducing the intensity of fireflies.

A.1 Regularization

Regularization of a BSDF involves approximating a narrow distribution with a function of wider support, known as a *mollifier*, which improves successful sample generation at the cost of increased bias. Kaplanyan & Dachsbacher (2013) introduce the concept of mollification and regularization to rendering, and explain the various ways that they can be applied. Of particular interest to us are spatial and angular regularization, which mollify BSDFs in the spatial and angular domains respectively, and selective regularization, which seeks to regularize only those BSDFs that would be most impactful. Since regularization results in a biased image, the error of which increases with the magnitude of mollification, a balance must be struck between optimizing for sampling efficiency versus minimizing bias. As a solution, Weier, Droske, Hanika, Weidlich & Vorba (2021) explore adaptively calculating a global regularization constant.

A.2 Virtual point lights

One method to cheaply approximate global illumination is to trace a small number of light paths and store the contribution at each vertex as a *virtual point light* (VPL), all of which are then sampled appropriately (Keller, 1997). In particular, given a light path $\bar{\mathbf{y}} = \overline{\mathbf{y}_0\mathbf{y}_1\cdots\mathbf{y}_k}$ where $\mathbf{y}_0$ is a light vertex, each path vertex $\mathbf{y}_i$ stores the following contribution:

$$\Phi(\mathbf{y}_j) = L_e(\mathbf{y}_0 \to \mathbf{y}_1)\prod_{i=2}^{j} f_r(\mathbf{y}_{i-2} \to \mathbf{y}_{i-1} \to \mathbf{y}_i)\prod_{i=1}^{j}\frac{G(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i)}{p_A(\mathbf{y}_i)} \quad 1 \le j \le k. \tag{A A.1}$$

If $\mathcal{P} = \{\mathbf{p}_j\}_j$ is the set of VPLs in the scene, then the outgoing radiance from $\mathbf{x}_1$ is simply the sum of the incoming contributions over $\mathcal{P}$:

$$L(\mathbf{x}_1) = \sum_{\mathbf{p}_j \in \mathcal{P}} f_r(\mathbf{p}_j \to \mathbf{x}_1 \to \mathbf{x}_0)G(\mathbf{p}_j \leftrightarrow \mathbf{x}_1)f_r(\mathbf{x}_3 \to \mathbf{p}_j \to \mathbf{x}_1)\Phi(\mathbf{p}_j). \tag{A A.2}$$

Assuming we sample VPLs uniformly so that $p(\bar{\mathbf{y}}) = 1/|\mathcal{P}|$, the radiance estimator is then

$$\hat{L}(\mathbf{x}_1) = \frac{|\mathcal{P}|}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_{2_j}) f_r(\mathbf{x}_{2_j} \to \mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_{2_j} \leftrightarrow \mathbf{x}_1) f_r(\mathbf{x}_3 \to \mathbf{x}_{2_j} \to \mathbf{x}_1). \tag{A A.3}$$

Note that this requires VPLs to store the position of $\mathbf{x}_3 = \mathbf{y}_{k-1}$ (or, alternatively, the incident radiance direction $\omega_i = \omega_{\mathbf{x}_{2_j} \to \mathbf{x}_3}$).

A.2.1 Bidirectional VPLs

Segovia, Iehl, Mitanchey & Péroche (2006) introduce a camera-traced formulation for VPLs. In light-traced VPLs, the density of each VPL is simply the light subpath pdf, which is in turn stored within Φ. However, this is not the case for camera-traced VPLs; instead, given a path $\bar{\mathbf{x}} = \overline{\mathbf{x}_1 \mathbf{x}_2}$ that terminates at a VPL located at $\mathbf{x}_2$, we have

$$p_A(\mathbf{x}_2) = \int_{\mathcal{M}_c} p(\mathbf{x}_1) p(\mathbf{x}_1 \to \mathbf{x}_2) \, \mathrm{d}A(\mathbf{x}_1) \,, \tag{A A.4}$$

where $\mathcal{M}_c$ is the set of all possible primary hit points. This integral can be estimated as

$$\hat{p}_A(\mathbf{x}_2) = \frac{1}{N} \sum_{i=1}^{N} p(\mathbf{x}_{1_i} \to \mathbf{x}_2). \tag{A A.5}$$

The rendering equation estimator from Equation (A A.3) then becomes

$$\hat{L}(\mathbf{x}_1) = \frac{|\mathcal{P}|}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_{2_j}) \frac{f_r(\mathbf{x}_{2_j} \to \mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_{2_j} \leftrightarrow \mathbf{x}_1) f_r(\mathbf{x}_3 \to \mathbf{x}_{2_j} \to \mathbf{x}_1)}{\hat{p}_A(\mathbf{x}_{2_j})}. \tag{A A.6}$$

Light-traced VPLs are generated without foreknowledge of the sensor position, which can lead to significant occlusion, which in turn increases the number of VPLs that must be generated. Furthermore, high-probability light paths will generate an excessive amount of VPLs, leading to undersampling of low-probability paths — usually the opposite of what we want. Li, Wang, Lin & Yuksel (2022) demonstrate how camera-traced VPLs can solve both of these issues by ensuring that all generated VPLs are visible to the sensor, and by varying the density of VPLs towards a more desirable distribution.

A.3 Virtual spherical lights

Because of their point-based nature, VPLs produce lighting singularities in glossy scenes, as reconnections with surfaces with small specular lobes tend to fail. VSLs (Hašan *et al.*, 2009) mitigate the singularities exhibited by VPLs by introducing a virtual spherical region that mollifies the connecting ray, as can be seen in Figure A.1. Given primary hit $\mathbf{x}_1$ and VSL located

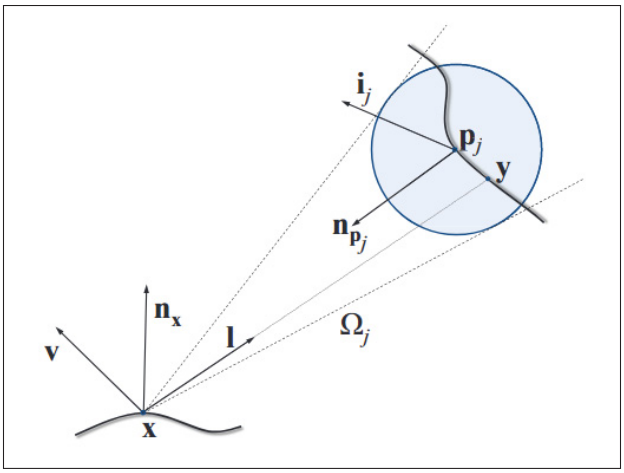


Figure-A A.1 VSLs and their accompanying definitions
Taken from Hašan *et al.* (2009)

at $\mathbf{p}_j$, we denote the incident direction at $\mathbf{p}_j$ as ω_i, the outgoing direction at $\mathbf{x}_1$ as ω_o, and let $\ell = \omega_{\mathbf{x}_1 \to \mathbf{p}_j}$ be the light direction; the outgoing radiance is then described by

$$L_{\mathbf{p}_j}(\mathbf{x}_1, \omega_o) = \frac{\Phi(\mathbf{p}_j)}{\pi r_j^2} V(\mathbf{x} \leftrightarrow \mathbf{p}_j) \int_{\Omega_j} f_r(\mathbf{x}_1, \omega_o, \ell) \left|\mathbf{n}_{\mathbf{x}_1} \cdot \ell\right| f_r(\text{trace}(\mathbf{x}_1, \ell), -\ell, \omega_i) \times \mathbf{1}[\left\|\mathbf{p}_j - \text{trace}(\mathbf{x}_1, \ell)\right\| < r_j] \, \mathrm{d}\ell \,. \tag{A A.7}$$

It is difficult to compute the indicator term $\mathbf{1}[\left\|\mathbf{p}_j - \text{trace}(\mathbf{x}_1, \ell)\right\| < r_j]$, which describes whether ℓ falls within the solid angle Ω_j subtended by the VSL, so we approximate it with $\left|\mathbf{n}_{\mathbf{p}_j} \cdot \ell\right|$:

$$L_{\mathbf{p}_j}(\mathbf{x}_1, \omega_o) \approx \frac{\Phi(\mathbf{p}_j)}{\pi r_j^2} V(\mathbf{x}_1 \leftrightarrow \mathbf{p}_j) \int_{\Omega_j} f_r(\mathbf{x}_1, \omega_o, \ell) \left|\mathbf{n}_{\mathbf{x}_1} \cdot \ell\right| f_r(\mathbf{p}_j, -\ell, \omega_i) \left|\mathbf{n}_{\mathbf{p}_j} \cdot \ell\right| \mathrm{d}\ell \,. \tag{A A.8}$$

This gives rise to the estimator

$$\hat{L}_{\mathbf{p}_j}(\mathbf{x}_1, \omega_o) \approx \frac{\Phi(\mathbf{p}_j)}{\pi r_j^2} V(\mathbf{x}_1 \leftrightarrow \mathbf{p}_j) \frac{1}{M} \sum_{m=1}^{M} \frac{f_r(\mathbf{x}_1, \omega_o, \ell_m) \left|\mathbf{n}_{\mathbf{x}_1} \cdot \ell_m\right| f_r(\mathbf{p}_j, -\ell_m, \omega_i) \left|\mathbf{n}_{\mathbf{p}_j} \cdot \ell_m\right|}{p_{\Omega_j}(\ell_m)} . \tag{A A.9}$$

Note that the sampled direction ℓ_m does not correspond to any intersection point in the above formulation; this is done to avoid performing an extra ray intersection query. However, letting $\mathbf{x}_3$ denote the first indirect bounce past the VSL, we can still (conceptually) convert this formulation

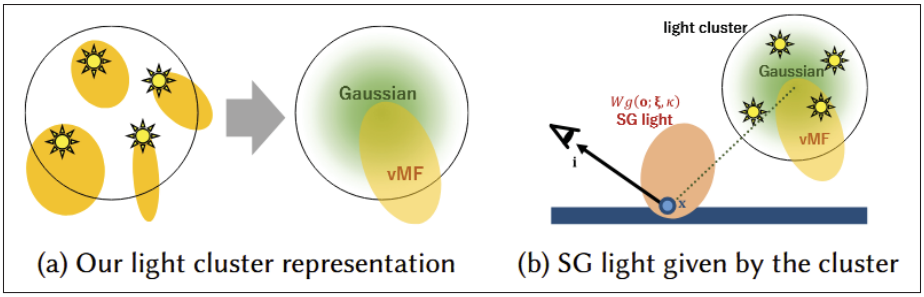


Figure-A A.2 A visualization of how VSGLs cluster VPLs
Taken from Tokuyoshi *et al.* (2024)

to area measure:

$$\hat{L}_{\mathbf{p}_j}(\mathbf{x}_1) \approx \frac{\Phi(\mathbf{p}_j)}{\pi r_j^2} V(\mathbf{x}_1 \leftrightarrow \mathbf{p}_j) \frac{1}{M} \sum_{m=1}^{M} \frac{f_r(\mathbf{p}_j \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{y}) f_r(\mathbf{x}_3 \rightarrow \mathbf{p}_j \rightarrow \mathbf{x}_1) \left|\mathbf{n}_{\mathbf{p}_j} \cdot \ell_m\right|}{p_A(\mathbf{y}_m)}. \tag{A A.10}$$

A.4 Virtual spherical Gaussian lights

Although VSLs drastically reduce the occurrence of lighting singularities, the additional round of sampling seen in Equations (A A.8) and (A A.9) makes them unsuitable for real-time rendering. Tokuyoshi (2015) hence introduce the concept of virtual spherical Gaussian lights (VSGLs), which bundle spatially close VPLs into a positional and directional spherical Gaussian approximation (Figure A.2). A spherical Gaussian generalizes the Gaussian distribution to the solid angle domain:

$$SG(\omega; \zeta, \lambda) = \exp(\lambda(\omega \cdot \zeta - 1)), \tag{A A.11}$$

where ζ and λ are the axis and sharpness of the SG respectively. In turn, the integral of an SG is

$$A(\lambda) = \int_{S^2} SG(\omega; \zeta, \lambda) \mathrm{d}\omega = \frac{2\pi}{\lambda}(1 - \exp(-2\lambda)). \tag{A A.12}$$

Hence, the normalized SG distribution $SG(\omega; \zeta, \lambda)/A(\lambda)$ has mean ζ and variance $1/\lambda$. The emitted radiance of the VSGL located at $\mathbf{p}_v$ is then

$$L(\mathbf{x}_1, \omega_o) = V(\mathbf{x}_1 \leftrightarrow \mathbf{p}_v) \frac{c_v}{2\pi\sigma_v^2} \exp(\lambda_{in} - \lambda_v - \lambda_\sigma) SG(\omega, \zeta_{in}, \lambda_{in}), \tag{A A.13}$$

where the unknowns in the above are computed as in Tokuyoshi (2015).

A.5 Method

With VSGLs, we can simply replace the incoming radiance term $L(\mathbf{x}, -\omega_i)$ with the emitted radiance of the VSGL. For each reservoir chosen for spatial reuse, we build a VSGL on-the-fly that contains its contribution stored as the luminance of the virtual light, the initial incident direction of the reconnection vertex as the SG's axis, and the roughness of the BSDF as the SG's sharpness as calculated in Tokuyoshi (2015). The spherical Gaussian then replaces the BSDF evaluation at the reconnection vertex, mollifying it and lowering the shift mapping failure rate.

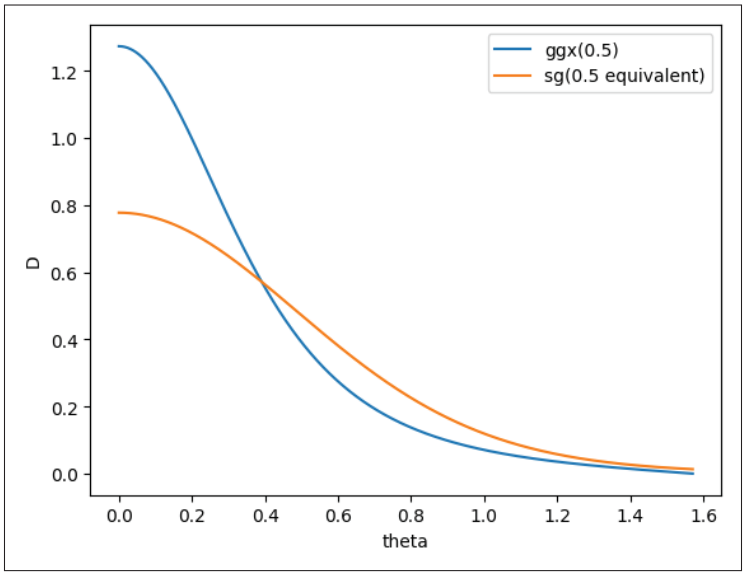


Figure-A A.3 Comparing the tail of the Trowbridge-Reitz distribution (Walter *et al.*, 2007) with the tail of the equivalent SG distribution. The x-axis is provided in radians

Figure A.3 compares the tail of the Trowbridge-Reitz distribution (Walter *et al.*, 2007), which is commonly used for microfacet BSDF, with that of the equivalent SG distribution. The SG distribution has heavier tails, which makes it a good candidate for mollifying BSDFs. Since the Trowbridge-Reitz distribution operates on the angular domain, we must parameterize the SG distribution accordingly in order to compare the two. In this case, the formula for an SG is

$$SG(\theta; \lambda) = \exp(\lambda(\cos\theta - 1)), \tag{A A.14}$$

and its integral is

$$\begin{aligned} A(\lambda) &= \int_{-\pi}^{\pi} SG(\theta;\lambda)\mathrm{d}\theta \\ &= \int_{-\pi}^{\pi} \exp(\lambda\cos\theta)\exp(-\lambda)\mathrm{d}\theta \\ &= \exp(-\lambda)\int_{-\pi}^{\pi} \exp(\lambda\cos\theta)\mathrm{d}\theta \\ &= 2\exp(-\lambda)\int_{0}^{\pi} \exp(\lambda\cos\theta)\mathrm{d}\theta \\ &= 2\pi\exp(-\lambda)I_0(\lambda), \end{aligned} \tag{A A.15}$$

where I_0 is the modified Bessel function of the first kind.

A.6 Results

Figure A.4 shows the outcome of applying our method to ReSTIR GI (Ouyang *et al.*, 2021). As the false-colour results show, the magnitude of the boils in ReSTIR GI is significantly reduced; this is supported by the corresponding decrease in the L^2 metric defined as

$$\mathrm{L}^2 = \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}}(\langle I_j\rangle - I_j)^2, \tag{A A.16}$$

where $\mathcal{J}$, I, and $\langle I\rangle$ are defined as in Equation (3.11). We choose the L^2 metric in this case as it is sensitive to outliers, and hence effectively highlights the magnitude reduction of the boiling artifacts. However, the output images on the top row show that the number of boils increases, though the overall intensity is lessened. Furthermore, introducing VSGLs results in bias from high-contribution samples being replaced with low-contribution samples.

A.7 Discussion

Our method is incomplete; the reason for the sample replacement that leads to the latent bias is unknown, and it is difficult to justify whether the benefits of reducing the peak magnitude of fireflies is worth the increased boiling that results. More experiments must be performed to determine how these changes impact real-time denoising algorithms (Schied *et al.*, 2017; Munkberg & Hasselgren, 2020), particularly whether the increased distribution of high-contribution samples allows for more efficient caustics rendering.

Currently, our method is only implemented for spatial reuse, though it should be simple to extend it to temporal reuse. Further extensions to our method involve using VSGLs to represent multiple world-space reservoirs (Boissé, 2021; Zhang & Wang, 2023), or storing the VSGLs in the reservoirs themselves, which would allow for direct application of Tokuyoshi *et al.* (2024)

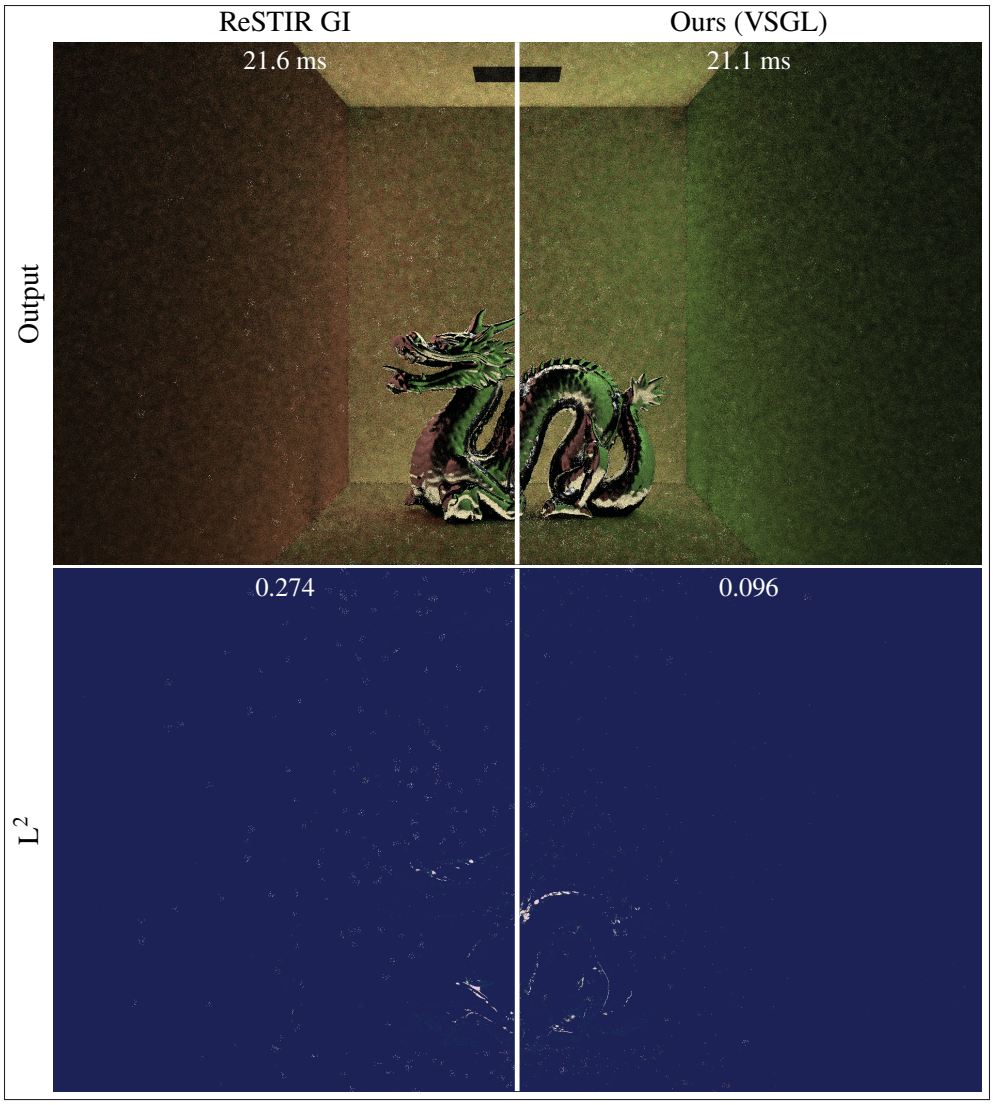


Figure-A A.4 Comparing the one-frame output of VSGLs with ReSTIR GI on the Cornell Box with Dragon scene

with regards to sample generation. Our mipmapped reservoir algorithm from Chapter 3 can also be modified to combine VSGLs over the spatial domain, since doing so only requires simple arithmetic on their parameters.

APPENDIX B

BOILING ARTIFACTS IN SPATIAL REUSE

As discussed in Chapter 2, fireflies in the original samples result in boiling artifacts after spatial reuse is applied. To explain why, suppose we have M candidate distributions but $N > M$ samples. Then

$$\langle \mathbb{E}[X \mid N] \rangle = \sum_{i=1}^{M} \frac{n_i m_i(x_i) f(x_i)}{p_i(x_i)}, \tag{A B.1}$$

where $\sum_{i=1}^{M} n_i = N$, the total number of samples taken. Note if $f(x_i) \gg 0$ is very large and/or $p_i(x_i)$ is very small for any i, then this estimator will have a noticeably significant value. Such scenarios correspond to low-probability events that in turn have a very large UCW; usually, these events stem from specular materials which have near-delta BSDFs. This is further compounded if the next bounce lands on an emissive surface, further increasing the contribution. Overall, we calculate the bias

$$\langle \mathbb{E}[X \mid N] \rangle - \mathbb{E}[X] = \sum_{i=1}^{M} \frac{m_i(x_i) f(x_i)}{p_i(x_i)} (n_i - 1). \tag{A B.2}$$

Given $N > M$, it is likely that $n_i > 1$ as well for each candidate distribution i, which means that the above value can be very large. Hence, any pixel that samples and uses an offending $f(x_i)$ will also have a large contribution, resulting in significant correlation which is visible as boiling.

APPENDIX C

PROOFS

C.1 Sampling distribution of weighted reservoir sampling

Let $X_1, X_2, \cdots, X_M$ be a set of samples with corresponding weights $w_1, w_2, \cdots, w_M$, so that the sampling probability of any X_i is $w_i / \sum_{k=1}^{M} w_k$. Let Y be the sample returned by weighted reservoir sampling. Then:

$$\begin{aligned} P(Y = X_i) &= P(X_i \text{ chosen}) P(X_{i+1} \text{ not chosen}) P(X_{i+2} \text{ not chosen}) \cdots P(X_M \text{ not chosen}) \\ &= \left(\frac{w_i}{\sum_{k=1}^{i} w_k} \right) \left(1 - \frac{w_{i+1}}{\sum_{k=1}^{i+1} w_k} \right) \left(1 - \frac{w_{i+2}}{\sum_{k=1}^{i+2} w_k} \right) \cdots \left(1 - \frac{w_M}{\sum_{k=1}^{M} w_k} \right) \\ &= \frac{w_i}{\cancel{\sum_{k=1}^{i} w_k}} \frac{\cancel{\sum_{k=1}^{i} w_k}}{\cancel{\sum_{k=1}^{i+1} w_k}} \frac{\cancel{\sum_{k=1}^{i+1} w_k}}{\cancel{\sum_{k=1}^{i+2} w_k}} \cdots \frac{\cancel{\sum_{k=1}^{M-1} w_k}}{\sum_{k=1}^{M} w_k} \\ &= \frac{w_i}{\sum_{k=1}^{M} w_k}, \end{aligned} \tag{A C.1}$$

which is the expected probability.

C.2 Bias of a parameterized unbiased estimator

Suppose $\langle X; y \rangle$ is an unbiased estimator of $f(X; y)$, where f is injective. Given an estimator $\langle Y \rangle$ for y, we have

$$\begin{aligned} \mathbb{E}[\langle X; \langle Y \rangle \rangle] &= \mathbb{E}\Big[\mathbb{E}[f(X; \langle Y \rangle)]\Big] \\ &= \mathbb{E}\left[\int f(x; \langle Y \rangle) p(x) \, \mathrm{d}x \right] \\ &= \int f(x; \mathbb{E}[\langle Y \rangle]) p(x) \, \mathrm{d}x \,, \end{aligned} \tag{A C.2}$$

Since f is injective, this is unbiased if and only if $\langle Y \rangle$ is unbiased; that is, $\mathbb{E}[\langle Y \rangle] = y$.

APPENDIX D

CODE LISTINGS

D.1 GPU-based SLIC cluster importance sampling

We present an algorithm for performing parallel importance sampling over SLIC clusters on the GPU, implemented in the Slang shading language (He, 2021). The basis of our algorithm is the `Find_Center_Association` function in gSLICr (Ren *et al.*, 2015), which we repurpose to find cells belonging to a given cluster by starting at the initial grid-based cluster and searching for the corresponding cell in all surrounding eight clusters; assuming one thread group per superpixel and one thread per cell, each thread thus reads nine reservoirs. We then perform a GPU prefix sum pass (Meister, Yoshimura & Kao, 2023) to compute and gather the RIS weights for each reservoir, followed by parallel importance sampling (Lu, Pacanowski & Granier, 2013) to conduct the resampling process and return a single output reservoir for the cluster. Notably, we do not use reservoir sampling, since doing so would require inter-thread synchronization.

Listing-A D.1 GPU-based SLIC importance resampling

```
static const uint gSpSize = SLIC_SPIXEL_SIZE;
static const uint gSpPixelCount = SLIC_SPIXEL_SIZE * SLIC_SPIXEL_SIZE;
static const uint gWaveLaneCount = 32;

struct SLICInfo
{
  uint numPixels;
};

StructuredBuffer<SLICInfo> gSpInfo;
Texture1D<uint> gIndices;

cbuffer
{
  uint2 gFrameDim;
  uint gFrameIndex;
};

struct Context
{
  float ws[9];
  float sum[9];
};

// Prefix sum

groupshared float s_prefixSumCache[gSpPixelCount / gWaveLaneCount];
groupshared float s_cellSumOffset;

void prefixSumOverCell(uint3 gID, uint3 tID, uint threadInd, uint
    spixelIndex, SLICInfo sp, out Context ctx)
```

```
{
  if (threadInd == 0)
  {
    s_cellSumOffset = 0;
  }

  [unroll]
  for (uint n = 0; n < 9; ++n)
  {
    let spixelOffset = int2(n % 3 - 1, n / 3 - 1);
    let pixel = gSpSize * (gID.xy + spixelOffset) + tID.xy;
    let r = getReservoir(pixel);

    var w = 0.f;
    if (sp.numPixels > 0 && all(pixel >= 0) && all(pixel < gFrameDim)
        && gIndices[pixel] == spixelIndex)
    {
      w = (1.f / (float)sp.numPixels) * r.r.targetDensity * r.r.W;
    }

    var prefixSum = WavePrefixSum(w);
    if (threadInd % gWaveLaneCount == gWaveLaneCount - 1)
    {
      s_prefixSumCache[threadInd / gWaveLaneCount] = w + prefixSum;
    }

    GroupMemoryBarrierWithGroupSync();

    // Add global prefix sum offset
    prefixSum += s_cellSumOffset;

    // Sum across waves
    if (threadInd < gSpPixelCount / gWaveLaneCount)
    {
      let waveSum = s_prefixSumCache[threadInd];
      let partialSum = WavePrefixSum(waveSum);
      s_prefixSumCache[threadInd] = partialSum;
    }

    GroupMemoryBarrierWithGroupSync();

    // Add cross-wave prefix sums to get cell-wise prefix sum
    prefixSum += s_prefixSumCache[threadInd / gWaveLaneCount];

    if (threadInd == gSpPixelCount - 1)
    {
      s_cellSumOffset = w + prefixSum;  // prefix sum for entire cell
    }

    ctx.ws[n] = w;
    ctx.sums[n] = prefixSum;
  }
```

```
82  }
83
84  // Parallel importance sampling
85
86  groupshared float s_u;
87  groupshared uint2 s_selectedPixel;
88
89  void parallelImportanceSample(uint3 gID, uint3 tID, uint threadInd,
        SLICInfo sp, Context ctx)
90  {
91    if (threadInd == 0)
92    {
93      s_u = sampleNext1D();
94      s_selectedPixel = sp.centre;
95    }
96
97    [unroll]
98    for (uint n = 0; n < 9; ++n)
99    {
100     let spixelOffset = int2(n % 3 - 1, n / 3 - 1);
101     let pixel = gSpSize * (gID.xy + spixelOffset) + tID.xy;
102     let lower = ctx.sums[n] / totalSum;
103     let upper = (ctx.sums[n] + ctx.ws[n]) / totalSum;
104     if (all(pixel >= 0) && all(pixel < gFrameDim) && lower <= s_u &&
            s_u < upper)
105     {
106       s_selectedPixel = pixel;
107     }
108   }
109 }
110
111 // 1 group per superpixel
112 [WaveSize(gWaveLaneCount)]
113 [numthreads(SLIC_SPIXEL_SIZE, SLIC_SPIXEL_SIZE, 1)]
114 void main(uint3 threadID: SV_DispatchThreadID, uint3 gID: SV_GroupID,
        uint3 tID: SV_GroupThreadID, uint threadInd: SV_GroupIndex)
115 {
116   if (any(gID.xy >= gMapDim))
117     return;
118
119   let spixelIndex = gID.y * gMapDim.x + gID.x;
120   let sp = gSpInfo[spixelIndex];
121   Context ctx;
122
123   // Prefix sum for each cell
124   prefixSumOverCell(gID, tID, threadInd, spixelIndex, sp, ctx);
125
126   GroupMemoryBarrierWithGroupSync();
127
128   // Normalize weights
129   let totalSum = s_cellSumOffset;
130
```

```
131     if (totalSum == 0)
132     {
133       gOutput[spixelIndex] = getReservoir(sp.centre);
134       return;
135     }
136 
137     // Parallel importance sampling
138     parallelImportanceSample(gID, tID, threadInd, sp, ctx);
139 
140     GroupMemoryBarrierWithGroupSync();
141 
142     // Write selected reservoir to output
143     if (threadInd == 0)
144     {
145       var rOut = getReservoir(s_selectedPixel);
146       rOut.r.W = rOut.r.targetDensity == 0 ? 0 : totalSum / rOut.r.
              targetDensity;
147       gOutput[spixelIndex] = rOut;
148     }
149   }
```

LIST OF REFERENCES

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274–2282. doi: 10.1109/TPAMI.2012.120.

Alber, L. D., Hanika, J. & Dachsbacher, C. (2025). Real-Time Markov Chain Path Guiding for Global Illumination and Single Scattering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 8(1), 15:1–15:18. doi: 10.1145/3728296.

Amazon Lumberyard. (2017, July). Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). Retrieved from: http://developer.nvidia.com/orca/amazon-lumberyard-bistro.

Andersson, P., Nilsson, J., Shirley, P. & Akenine-Möller, T. (2021, May). Visualizing Errors in Rendered High Dynamic Range Images. *Eurographics Short Papers*. doi: 10.2312/egs.20211015.

Bartels, P. & Harada, T. (2022, December). Combining GPU Tracing Methods within a Single Ray Query. *SIGGRAPH Asia 2022 Technical Communications*, pp. 17:1–17:4. doi: 10.1145/3550340.3564231.

Bitterli, B. (2016). Rendering resources. Retrieved from: https://benedikt-bitterli.me/resources/.

Bitterli, B. (2022). *Correlations and Reuse for Fast and Accurate Physically Based Light Transport*. (Ph.D. thesis, Dartmouth College).

Bitterli, B., Wyman, C., Pharr, M., Shirley, P., Lefohn, A. & Jarosz, W. (2020). Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics*, 39(4), 148:1–148:17. doi: 10.1145/3386569.3392481.

Blinn, J. F. (1989). Jim Blinn's Corner: Return of the Jaggy. *IEEE Computer Graphics and Applications*, 9(2), 82–89. doi: 10.1109/38.19054.

Boissé, G. (2021, December). WORLD-SPACE SPATIOTEMPORAL RESERVOIR REUSE FOR RAY-TRACED GLOBAL ILLUMINATION. *SIGGRAPH Asia 2021 Technical Communications*, pp. 22:1–22:4. doi: 10.1145/3478512.3488613.

Burley, B. & Walt Disney Animation Studios. (2012). Physically-Based Shading at Disney. *ACM SIGGRAPH 2012 Courses*. Retrieved from: https://disneyanimation.com/publications/physically-based-shading-at-disney/.

Bálint, M., Salaün, C., Seidel, H.-P. & Myszkowski, K. (2026). Forget Superresolution, Sample Adaptively (when Path Tracing). *ACM Transactions on Graphics*, 45(4), 90:1–90:19. doi: 10.1145/3811377.

Chang, W., Sivaram, V., Nowrouzezahrai, D., Hachisuka, T., Ramamoorthi, R. & Li, T.-M. (2023, July). Parameter-space ReSTIR for Differentiable and Inverse Rendering. *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 18:1–18:10. doi: 10.1145/3588432.3591512.

Chao, M. T. (1982). A general purpose unequal probability sampling plan. *Biometrika*, 69(3), 653–656. doi: 10.1093/biomet/69.3.653.

Chermain, X., Sauvage, B., Dischler, J.-M. & Dachsbacher, C. (2021). Importance Sampling of Glittering BSDFs based on Finite Mixture Distributions. *Eurographics Symposium on Rendering*, pp. 45–53. doi: 10.2312/SR.20211289.

Crassin, C., Neyret, F., Sainz, M., Green, S. & Eisemann, E. (2011). Interactive Indirect Illumination Using Voxel Cone Tracing. *Computer Graphics Forum*, 30(7), 1921–1930. doi: 10.1111/j.1467-8659.2011.02063.x.

Dachsbacher, C. & Stamminger, M. (2005, April). Reflective shadow maps. *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pp. 203–231. doi: 10.1145/1053427.1053460.

Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B. & Novák, J. (2014). Scalable Realistic Rendering with Many-Light Methods. *Computer Graphics Forum*, 33(1), 88–104. doi: 10.1111/cgf.12256.

Davis, P. J. & Rabinowitz, P. (2007). *Methods of numerical integration*. Courier Corporation.

Dittebrandt, A., Hanika, J. & Dachsbacher, C. (2020). Temporal Sample Reuse for Next Event Estimation and Path Guiding for Real-Time Path Tracing. *Eurographics Symposium on Rendering*, pp. 39–51. doi: 10.2312/SR.20201135.

Dittebrandt, A., Schüßler, V., Hanika, J., Herholz, S. & Dachsbacher, C. (2023). Markov Chain Mixture Models for Real-Time Direct Illumination. *Computer Graphics Forum*, 42(4), e14881. doi: 10.1111/cgf.14881.

Dupuy, J. & Benyoub, A. (2023). Sampling Visible GGX Normals with Spherical Caps. *Computer Graphics Forum*, 42(8), e14867. doi: 10.1111/cgf.14867.

Gauthier, A., Faury, R., Levallois, J., Thonat, T., Thiery, J.-M. & Boubekeur, T. (2022). MIPNet: Neural Normal-to-Anisotropic-Roughness MIP Mapping. *ACM Transactions on Graphics*, 41(6), 246:1–246:12. doi: 10.1145/3550454.3555487.

Green, R. (2003). Spherical Harmonic Lighting: The Gritty Details. *Game Developers Conference*.

Gregory, J. (2026). *Game Engine Architecture: Volume II, Graphics, Motion, and Sound* (ed. 4). Boca Raton: CRC Press. doi: 10.1201/9781003683605.

Hansen, N. (2023, March). The CMA Evolution Strategy: A Tutorial. arXiv. Retrieved on 2026-06-17 from: http://arxiv.org/abs/1604.00772.

Hašan, M., Křivánek, J., Walter, B. & Bala, K. (2009). Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics*, 28(5), 1–6. doi: 10.1145/1618452.1618489.

He, Y. (2021). *Slang — A Shader Compilation System for Extensible, Real-Time Shading*. (Ph.D. thesis, Carnegie Mellon University).

Heckbert, P. (1989). *Fundamentals of Texture Mapping and Image Warping*. (Ph.D. thesis, UC Berkeley).

Heckbert, P. S. (1986). Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11), 56–67. doi: 10.1109/MCG.1986.276672.

Hedstrom, T., Kettunen, M., Lin, D., Wyman, C. & Li, T.-M. (2025). ReSTIR BDPT: Bidirectional ReSTIR Path Tracing with Caustics. *ACM Transactions on Graphics*, 44(5), 169:1–169:16. doi: 10.1145/3744898.

Hedstrom, T., Kettunen, M., Lin, D., Wyman, C. & Li, T.-M. (2026). Stochastic Pairwise MIS for Unbiased Large-Kernel Reuse in Real Time. *Computer Graphics Forum*, 45(2), e70391. doi: 10.1111/cgf.70391.

Heitz, E. & d'Eon, E. (2014). Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals. *Computer Graphics Forum*, 33(4), 103–112. doi: 10.1111/cgf.12417.

Heitz, E. (2018). Sampling the GGX Distribution of Visible Normals. *Journal of Computer Graphics Techniques (JCGT)*, 7(4), 1–13. Retrieved from: http://jcgt.org/published/0007/04/01/.

Heitz, E., Hanika, J., d'Eon, E. & Dachsbacher, C. (2016). Multiple-scattering microfacet BSDFs with the Smith model. *ACM Transactions on Graphics*, 35(4), 58:1–58:14. doi: 10.1145/2897824.2925943.

Hofmann, N., Hasselgren, J., Clarberg, P. & Munkberg, J. (2021). Interactive Path Tracing and Reconstruction of Sparse Volumes. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1), 5:1–5:19. doi: 10.1145/3451256.

Holl, S., Singh, G. & Seidel, H.-P. (2026). Diffusion Restore: Real-Time Markov Chain Monte Carlo Light Transport. arXiv. Retrieved on 2026-05-15 from: https://arxiv.org/abs/2605.08916.

Hong, P., Duan, M., Wang, B., Yuksel, C., Zeltner, T. & Lin, D. (2025, December). Sample Space Partitioning and Spatiotemporal Resampling for Specular Manifold Sampling. *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 13:1–13:10. doi: 10.1145/3757377.3763927.

Hua, B., Gruson, A., Petitjean, V., Zwicker, M., Nowrouzezahrai, D., Eisemann, E. & Hachisuka, T. (2019). A Survey on Gradient-Domain Rendering. *Computer Graphics Forum*, 38(2), 455–472. doi: 10.1111/cgf.13652.

Jakob, W. & Marschner, S. (2012). Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics*, 31(4), 58:1–58:13. doi: 10.1145/2185520.2185554.

Jensen, H. W. (1996). Global Illumination using Photon Maps. *Rendering Techniques '96*, (Eurographics), 21–30. doi: 10.1007/978-3-7091-7484-5_3.

Junkins, O., Kettunen, M., Lin, D., Ramamoorthi, R. & Wyman, C. (2026). Compatibility-Guided Neighbor Selection for ReSTIR. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 9(4), 52:1–52:16. doi: 10.1145/3820024.

Kajiya, J. T. (1986). The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4), 143–150. doi: 10.1145/15886.15902.

Kallweit, S., Clarberg, P., Kolb, C., Davidovič, T., Yao, K.-H., Foley, T., He, Y., Wu, L., Chen, L., Akenine-Möller, T., Wyman, C., Crassin, C. & Benty, N. (2022, August). The Falcor Rendering Framework. Retrieved from: https://github.com/NVIDIAGameWorks/Falcor.

Kaplanyan, A. S. & Dachsbacher, C. (2013). Path Space Regularization for Holistic and Robust Light Transport. *Computer Graphics Forum*, 32(2pt1), 63–72. doi: 10.1111/cgf.12026.

Kelemen, C., Szirmay-Kalos, L., Antal, G. & Csonka, F. (2002). A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum*, 21(3), 531–540. doi: 10.1111/1467-8659.t01-1-00703.

Keller, A. (1997). Instant radiosity. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 49–56. doi: 10.1145/258734.258769.

Kern, R., Brüll, F. & Grosch, T. (2024). ReSTIR FG: Real-Time Reservoir Resampled Photon Final Gathering. *Eurographics Symposium on Rendering*, pp. 13 pages. doi: 10.2312/SR.20241155.

Kettunen, M., Manzi, M., Aittala, M., Lehtinen, J., Durand, F. & Zwicker, M. (2015). Gradient-domain path tracing. *ACM Transactions on Graphics*, 34(4), 123:1–123:13. doi: 10.1145/2766997.

Kettunen, M., Lin, D., Ramamoorthi, R., Bashford-Rogers, T. & Wyman, C. (2023, December). Conditional Resampled Importance Sampling and ReSTIR. *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–11. doi: 10.1145/3610548.3618245.

Kontoghiorghes, E. J. (2005). *Handbook of parallel computing and statistics*. CRC Press.

Lawrence, J., Rusinkiewicz, S. & Ramamoorthi, R. (2004). Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics*, 23(3), 496–505. doi: 10.1145/1015706.1015751.

Li, T., Wang, W., Lin, D. & Yuksel, C. (2022). Virtual Blue Noise Lighting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(3), 23:1–23:26. doi: 10.1145/3543872.

Li, T.-M. (2019). *Differentiable Visual Computing*. (Ph.D. thesis, Massachusetts Institute of Technology).

Lin, D. & Yuksel, C. (2020). Real-Time Stochastic Lightcuts. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1), 5:1–5:18. doi: 10.1145/3384543.

Lin, D., Wyman, C. & Yuksel, C. (2021). Fast volume rendering with spatiotemporal reservoir resampling. *ACM Transactions on Graphics*, 40(6), 1–18. doi: 10.1145/3478513.3480499.

Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C. & Wyman, C. (2022). Generalized resampled importance sampling: foundations of ReSTIR. *ACM Transactions on Graphics*, 41(4), 75:1–75:23. doi: 10.1145/3528223.3530158.

Lin, D., Kettunen, M. & Wyman, C. (2026). ReSTIR PT Enhanced: Algorithmic Advances for Faster and More Robust ReSTIR Path Tracing. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 9(1), 13:1–13:19. doi: 10.1145/3804494.

Liu, J., Lin, D., Kettunen, M., Wyman, C. & Ramamoorthi, R. (2025, August). Reservoir Splatting for Temporal Path Resampling and Motion Blur. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 35:1–35:11. doi: 10.1145/3721238.3730646.

Lu, H., Chang, W., Hedstrom, T. & Li, T.-M. (2024). Real-Time Path Guiding Using Bounding Voxel Sampling. *ACM Transactions on Graphics*, 43(4), 125:1–125:14. doi: 10.1145/3658203.

Lu, H., Pacanowski, R. & Granier, X. (2013). Real-Time Importance Sampling of Dynamic Environment Maps. *Eurographics 2013 - Short Papers*, pp. 65–68. doi: 10.2312/conf/EG2013/short/065-068.

Luksch, C., Wimmer, M. & Schwärzler, M. (2019, May). Incrementally baked global illumination. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 4:1–4:10. doi: 10.1145/3306131.3317015.

Majercik, Z., Guertin, J.-P., Nowrouzezahrai, D. & McGuire, M. (2019). Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields. *Journal of Computer Graphics Techniques*, 8(2), 1–30. Retrieved from: https://jcgt.org/published/0008/02/01/.

Meister, D., Yoshimura, A. & Kao, C.-C. (2023). GPU Programming Primitives for Computer Graphics. *ACM SIGGRAPH Asia 2023 Courses*, (SIGGRAPH Asia 2023), 9:1–9:81. doi: 10.1145/3610538.3614632.

Munkberg, J. & Hasselgren, J. (2020). Neural Denoising with Layer Embeddings. *Computer Graphics Forum*, 39(4), 1–12. doi: 10.1111/cgf.14049.

Müller, T., Gross, M. & Novák, J. (2017). Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum*, 36(4), 91–100. doi: 10.1111/cgf.13227.

Nehab, D. & Hoppe, H. (2014). A Fresh Look at Generalized Sampling. *Foundations and Trends in Computer Graphics and Vision*, 8(1), 1–84. doi: 10.1561/0600000053.

Ouyang, Y., Liu, S., Kettunen, M., Pharr, M. & Pantaleoni, J. (2021). ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Computer Graphics Forum*, 40(8), 17–29. doi: 10.1111/cgf.14378.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*. Retrieved from: https://artowen.su.domains/mc/.

Pharr, M. (2023). PBRT v4 Scenes. Retrieved from: https://github.com/mmp/pbrt-v4-scenes.

Pharr, M., Jakob, W. & Humphreys, G. (2017). *Physically based rendering: from theory to implementation* (ed. 3). Cambridge, MA: Morgan Kaufmann Publishers/Elsevier.

Pharr, M., Wronski, B., Salvi, M. & Fajardo, M. (2024). Filtering After Shading With Stochastic Texture Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1), 14:1–14:20. doi: 10.1145/3651293.

Ren, C. Y., Prisacariu, V. A. & Reid, I. D. (2015). gSLICr: SLIC superpixels at over 250Hz. arXiv. Retrieved on 2025-10-24 from: https://arxiv.org/abs/1509.04232.

Salaün, C., Balint, M., Belcour, L., Heitz, E., Singh, G. & Myszkowski, K. (2025, August). Histogram Stratification for Spatio-Temporal Reservoir Sampling. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 83:1–83:10. doi: 10.1145/3721238.3730723.

Sawhney, R., Lin, D., Kettunen, M., Bitterli, B., Ramamoorthi, R., Wyman, C. & Pharr, M. (2023). Decorrelating ReSTIR Samplers via MCMC Mutations. *ACM Transactions on Graphics*, 43(1), 10:1–10:15. doi: 10.1145/3629166.

Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C. R. A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A. & Salvi, M. (2017, July). Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. *Proceedings of High Performance Graphics*, pp. 2:1–2:12. doi: 10.1145/3105762.3105770.

Segovia, B., Iehl, J. C., Mitanchey, R. & Péroche, B. (2006). Bidirectional Instant Radiosity. *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, pp. 389–397.

Sugimoto, R., Chen, T., Jiang, Y., Batty, C. & Hachisuka, T. (2023). A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Transactions on Graphics*, 42(4), 81:1–81:16. doi: 10.1145/3592109.

Talbot, J., Cline, D. & Egbert, P. (2005). Importance Resampling for Global Illumination. *Eurographics Symposium on Rendering (2005)*, pp. 139–146. doi: 10.2312/EGWR/EGSR05/139-146.

Tevs, A., Ihrke, I. & Seidel, H.-P. (2008, February). Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pp. 183–190. doi: 10.1145/1342250.1342279.

Toksvig, M. (2005). Mipmapping Normal Maps. *Journal of Graphics Tools*, 10(3), 65–71. doi: 10.1080/2151237X.2005.10129203.

Tokuyoshi, Y. (2015). Virtual Spherical Gaussian Lights for Real-time Glossy Indirect Illumination. *Computer Graphics Forum*, 34(7), 89–98. doi: 10.1111/cgf.12748.

Tokuyoshi, Y., Ikeda, S., Kulkarni, P. & Harada, T. (2024, December). Hierarchical Light Sampling with Accurate Spherical Gaussian Lighting. *SIGGRAPH Asia 2024 Conference Papers*, pp. 82:1–82:11. doi: 10.1145/3680528.3687647.

Veach, E. (1997). *Robust Monte Carlo Methods for Light Transport Simulation*. (Ph.D. thesis, Stanford University).

Veach, E. & Guibas, L. J. (1995). Optimally combining sampling techniques for Monte Carlo rendering. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, pp. 419–428. doi: 10.1145/218380.218498.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1), 37–57. doi: 10.1145/3147.3165.

Walter, B., Marschner, S. R., Li, H. & Torrance, K. E. (2007). Microfacet Models for Refraction through Rough Surfaces. *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, (EGSR'07), 195–206. doi: 10.2312/EGWR/EGSR07/195-206.

Wang, B., Jin, W., Fan, J., Yang, J., Holzschuch, N. & Yan, L.-Q. (2022). Position-free multiple-bounce computations for smith microfacet BSDFs. *ACM Transactions on Graphics*, 41(4), 134:1–134:14. doi: 10.1145/3528223.3530112.

Weier, P., Droske, M., Hanika, J., Weidlich, A. & Vorba, J. (2021). Optimised Path Space Regularisation. *Computer Graphics Forum*, 40(4), 139–151. doi: 10.1111/cgf.14347.

West, R., Georgiev, I., Gruson, A. & Hachisuka, T. (2020). Continuous multiple importance sampling. *ACM Transactions on Graphics*, 39(4), 136:1–136:12. doi: 10.1145/3386569.3392436.

Williams, L. (1983). Pyramidal parametrics. *ACM SIGGRAPH Computer Graphics*, 17(3), 1–11. doi: 10.1145/964967.801126.

Wronski, B., Pharr, M. & Akenine-Möller, T. (2025). Improved Stochastic Texture Filtering Through Sample Reuse. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 8(1), 14:1–14:19. doi: 10.1145/3728292.

Wyman, C. & Panteleev, A. (2021). Rearchitecting Spatiotemporal Resampling for Production. *Proceedings of the Conference on High-Performance Graphics*, pp. 23–41. doi: 10.2312/hpg.20211281.

Wyman, C., Kettunen, M., Lin, D., Bitterli, B., Yuksel, C., Jarosz, W. & Kozlowski, P. (2023, July). A Gentle Introduction to ReSTIR Path Reuse in Real-Time. *ACM SIGGRAPH 2023 Courses*, pp. 1–38. doi: 10.1145/3587423.3595511.

Xu, X., Wang, L. & Wang, B. (2023). Efficient Caustics Rendering via Spatial and Temporal Path Reuse. *Computer Graphics Forum*, 42(7), e14975. doi: 10.1111/cgf.14975.

Zeng, Z., Kettunen, M., Wyman, C., Wu, L., Ramamoorthi, R., Yan, L.-Q. & Lin, D. (2025, December). ReSTIR PG: Path Guiding with Spatiotemporally Resampled Paths. *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 12:1–12:11. doi: 10.1145/3757377.3763813.

Zhang, H. & Wang, B. (2023). World-Space Spatiotemporal Path Resampling for Path Tracing. *Computer Graphics Forum*, 42(7), e14974. doi: 10.1111/cgf.14974.

Zhang, S., Lin, D., Kettunen, M., Yuksel, C. & Wyman, C. (2024). Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing. *ACM Transactions on Graphics*, 43(4), 98:1–98:13. doi: 10.1145/3658210.

Zhang, S., Lin, D., Wyman, C. & Yuksel, C. (2025). Many-Light Rendering Using ReSTIR-Sampled Shadow Maps. *Computer Graphics Forum*, 44(2), e70059. doi: 10.1111/cgf.70059.

Zheng, Q. & Zwicker, M. (2019). Learning to Importance Sample in Primary Sample Space. *Computer Graphics Forum*, 38(2), 169–179. doi: 10.1111/cgf.13628.