HIGH-QUALITY SAMPLING FOR COMPLEX EFFECTS IN REAL-TIME RAY TRACING

by

Daqi Lin

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

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

May 2022

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Daqi Lin**

has been approved by the following supervisory committee members:

Cem Yuksel,	Chair(s)	**14 April 2022** Date Approved
Erik L. Brunvand,	Member	**8 April 2022** Date Approved
Daniel M. Kopta,	Member	**8 April 2022** Date Approved
Chris Wyman,	Member	**6 April 2022** Date Approved
Larry Seiler,	Member	**8 April 2022** Date Approved

by **Mary Hall**, Chair/Dean of

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

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

ABSTRACT

This dissertation proposes high-quality sampling methods for real-time rendering of complex effects with ray tracing. Real-time ray tracing significantly improves the visual quality in real-time rendering applications in recent years. However, the application is mostly on simple effects like reflection and shadow, which require few samples. To render complex effects like global illumination and volumetric multiple scattering, it is critical for the sampling algorithm to be intelligent to avoid generating a large number of samples to achieve acceptable quality. In this dissertation, we develop an algorithm to combine rasterization and stochastic ray tracing to render shadowed illumination from a large number of virtual point lights. We then introduce a light tree sampling algorithm to efficiently estimate illumination in general many light scenarios in fully dynamic scenes. Moreover, we propose a real-time volume rendering method based on reservoir resampling that can render heterogeneous volume with multiple scattering in dynamic, complex scenes. We further generalize the resampled importance sampling theory to support more robust spatiotemporal reuse to dramatically improve the real-time sampling quality of difficult paths in global illumination.

For my parents.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Most traditional real-time rendering techniques rely on precomputation for complex lighting effects. For example, lightmap or irradiance volumes are baked to achieve diffuse global illumination in real-time. This precomputation approach limits the completeness, interactivity, and accuracy of the lighting effects. Recent advances in ray tracing hardware [13] and algorithms and the invention of high-quality denoisers [2, 14] introduce a paradigm shift from precomputed lighting to sampled lighting. A limited number of rays can be generated per frame to render simple effects like specular reflection, ambient occlusion, or soft shadow reasonably well after denoising [15]. However, using the same brute force Monte Carlo sampling leads to very high variance for complex effects like global illumination, which cannot be denoised well. In this dissertation, we propose several methods that improve the quality of sampling in real-time rendering for various complex effects. These algorithms use GPU-friendly simple data structures to minimize the performance overhead of sampling. We first explore the combination of rasterization and ray tracing to produce a low-noise, biased illumination estimation from many lights to approximate diffuse-dominant global illumination. Then we move to a completely sampling-based method that uses spatial tree structures to produce unbiased estimates for many light illumination. Later, we explore a recently introduced resampling theory based on spatiotemporal reuse to design a high-quality sampling algorithm for real-time volume rendering in complex environments. Finally, we generalize that resampling theory to fundamentally improve its quality and robustness to handle more complex effects in global illumination.

1.1 Contributions

The key contributions of this dissertation are listed as follows:

- We present an efficient method for lighting grid hierarchy [16] construction, rendering, and shadow sampling on the GPU, using a hybrid ray tracing-rasterization approach (Chapter 3). This allows rendering high-quality **diffuse-dominant global illumination** in complex scenes using many virtual lights, including dynamic lighting and dynamic geometry at real-time frame rates.

- We propose perfect balanced light trees that allows extremely fast construction and high traversal performance (Chapter 4). Combining our tree with stochastic lightcuts [4], we provide a high-performance solution for **general many-light rendering with arbitrary light types**.

- We equip ReSTIR [7] with the ability to sample complex path spaces and use it as an efficient importance sampling estimator for the volumetric path integral (Chapter 5). With optimized path space transmittance estimates and a novel temporal reprojection method, we achieve **low-noise, interactive volumetric rendering** with arbitrary dynamic lighting, including volumetric emission, and maintain interactive performance even on high-resolution volumes.

- We introduce generalized resampled importance sampling (GRIS) as a new theoretical framework that lifts the independent and identically distributed (i.i.d.) assumption of RIS [17] (Chapter 6). Besides providing a theoretical foundation for ReSTIR, our theory deepens the understanding of convergence and guides the design of advanced shift mappings for effective reuse across different sampling domains. This allows us to develop ReSTIR Path Tracing, a robust, unbiased light transport algorithm that can handle even **very complex lighting scenarios** (e.g., caustics) while remaining fully amenable to efficient GPU parallelization and real-time use. The algorithm can also be repurposed for offline rendering, providing superior convergence to unidirectional path tracing and prior work in path reuse.

1.2 Dissertation Statement

High-quality sampling, i.e., sampling methods that generate low-noise results with a low sample count, is important for synthesizing complex effects like many-light rendering, global illumination, and volume rendering in real-time rendering. Specifically, we can develop sampling algorithms that leverage the ray tracing capability of GPUs to produce real-time rendering results of these complex lighting effects with low cost and high quality, which is hard to achieve by naive sampling algorithms.

1.3 Dissertation Organization

This dissertation is structured as follows:

In Chapter 2, we review the prior work of many light rendering, real-time global illumination, volume rendering, and path reuse to provide background for the topics in later chapters.

In Chapter 3, we introduce *Real-Time Rendering with Lighting Grid Hierarchy*, a biased solution for diffuse global illumination from a large number of virtual point lights.

In Chapter 4, we introduce *Real-Time Stochastic Lightcuts*, a general many-light rendering method that features perfectly balanced light trees.

In Chapter 5, we introduce *Fast Volume Rendering with Spatiotemporal Reservoir Resampling* that allows volume rendering with complex effects like multiple scattering and self-emission to be rendered with arbitrary dynamic lighting and volume animation, while significantly outperforming the state-of-art path tracing approaches.

In Chapter 6, we introduce *Generalized Resampled Importance Sampling* (GRIS) that generalizes the theory of resampled importance sampling (RIS) and provides a foundation for ReSTIR. With the theory, we develop ReSTIR Path Tracing (ReSTIR PT), an unbiased global illumination method that better handles complex light transport and is robust enough to be even used for offline rendering.

Finally, we conclude in Chapter 7.

CHAPTER 2

BACKGROUND

Sampling, importance sampling, and sample reuse are key to modern renderers and form a substantial body of research in graphics, since they are crucial to efficiently render complex effects we study in this dissertation, including many light direct illumination, volume rendering, and general global illumination. A common problem for these effects is that efficiently sampling from the related integrals is difficult. By writing these integrals into a common form $\int_\Omega f(x)dx$, where Ω is the sampling domain and x is the variable (e.g., a scattering direction, a collision distance, a position on emissive surfaces) we want to sample, the N-sample Monte Carlo estimator is expressed as

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} .$$

We would like $p(x) \approx f(x)/\int_\Omega f(x)dx$ to minimize the variance of importance sampling. Finding a good $p(x)$ is difficult due to the unknown (before the sample is evaluated) quantities in the integrand, like visibility and transmittance. In many cases, $f(x)$ is defined in a recursive way, requiring nested integrals, making the shape of the integral difficult to predict. Even with known integrands, it is often difficult to find its (or its approximation's) integral in a closed form, making the CDF inversion method infeasible.

Numerous offline rendering methods exist to reduce variance in the sampling process. When targeting real-time, even with hardware ray tracing, improving quality at iso-performance is insufficient; results with low sample budgets need denoising [18, 19, 20, 2]. However, denoising inputs with insufficient sampling quality leads to low-frequency noise, blurring, and a lack of temporal stability. Therefore, it is important to develop real-time sampling methods with high quality.

The methods we propose in this dissertation provide high-quality real-time sampling. We define easy-to-sample $p(x)$ using tree structures (Chapter 4) or generating samples with good $p(x)$ based on resampling, whose efficiency comes from the reuse of lighting approximation (Chapter 3), or sampling distribution (Chapter 5, 6). The methods presented in this dissertation fit into a rich history of path reuse [11, 21], ratio estimators [22, 23], light importance sampling [24, 25], and various structures to accelerate sample lookups [26, 27, 3]. While also being variance reduction methods, the methods we propose respect real-time GPU rendering constraints with cheap data structures, low computational overhead, quick response to lighting or geometry change, and the ability to reach low error with low sample count (potentially sacrificing long-term convergence).

Having a unified solution that works for all complex effects with these constraints is difficult. Generally, we need to take advantage of the domain knowledge and combine ideas from state-of-art real-time rendering methods. In Section 2.1, Section 2.2, and Section 2.3, we review prior work on the rendering of the complex effects we study in the dissertation. Besides introducing sampling-based methods, we also introduce real-time or offline methods based on approximation and precomputation to provide the readers with an overview of the state of the art. Details about directly related works (e.g., stochastic lightcuts [4]) will be introduced in later chapters. The later part of the dissertation builds on path reuse and sample reuse. We review the background of path reuse in Section 2.4.

2.1 Many-Lights Rendering

The many-lights problem received considerable attention in computer graphics [28], starting with ordering lights based on their contributions [29], stochastic sampling [24], light clustering using octrees [30], and precomputed visibility culling [31]. The introduction of virtual point lights (VPLs) for approximating global illumination [32] has drawn more research interests into the many-lights problem. We review the prior work for VPL rendering in Chapter 3. The *lightcuts* method [26] provides a highly efficient scalable solution to the many lights problem by forming a binary light tree from the light sources. For each point in the scene, where lighting is evaluated, the lightcuts method picks a *cut* through the light tree and only computes the representative lights of the internal light tree nodes above this cut. It is also possible to avoid computing a cut for each pixel using reconstruction cut [26] or lightcut interpolation [33]. The extensions of lightcuts address high-dimensional integrations [34], progressive GPU implementation [35], bidirectional sampling [36], and out-of-core GPU implementation for large scenes [37].

The unbiasedness of lightcuts is achieved by randomizing the tree building. However, in one rendering, the correlation of pixel illumination estimates can cause distracting artefacts with insufficient cut size due to using a common set of lights for a large number of pixels. Recent methods use light tree for independent sampling to overcome the correlation. Vévoda and Křivánek [38] used the clusters formed by lightcuts for adaptive importance sampling of direct illumination instead of using them directly as the illumination approximation. A following work [39] uses Bayesian online regression to learn the light selection probability distributions for the light clusters. Keller et al. [40] added additional node information into the light tree, where each node stores directionally varying light intensities, instead of a single flux. Adaptive tree splitting [41] introduced a light bounding volume hierarchy (BVH) with a technique that splits light tree traversal based on the cluster variance. Yuksel [4] introduced stochastic lightcuts which provides low noise results using much fewer samples than other methods. With the advancement of high-quality denoisers and ray tracing capable GPUs, it is now possible to implement Monte Carlo sampling algorithms on the GPU using limited samples. Moreau and Clarberg [42] presented a version of adaptive tree splitting [41] for real-time rendering, and Moreau et al. [3] introduced a two-level light BVH builder for dynamic scenes. We extend stochastic lightcuts with perfect binary trees to maximize its efficiency for real-time rendering in Chapter 4.

An alternative solution to the many-lights problem is to form a lighting matrix and approximate its solution [43], known as matrix row-column sampling (MRCS). The extensions of this approach include a method for handling glossy surfaces [44], reducing flickering by processing animated sequences [45], and using cuts [46] or a reduced matrix [47] for accelerating the computation.

The lighting grid hierarchy method [16] introduces a temporally coherent approximation of a large number of virtual point lights for rendering self-illuminating volumes by using multiple representations of the illumination at different resolutions. We extend this method in Chapter 3 by providing a GPU-friendly variant that is suitable for real-time rendering with many lights.

2.2 Global Illumination

General global illumination includes all possible types of light paths. Offline rendering usually solves global illumination by path tracing and photon mapping. We refer the readers to the PBRT book [48] for a survey. In a real-time rendering context, many path types are too difficult to handle by common methods. Most real-time global illumination algorithm limits themselves to solve mostly diffuse (or slightly glossy) indirect illumination. Precomputation is popular in game engines for approximating global illumination in mostly static lighting environment. Lightmaps or light probes (irradiance volume) [49] are most effective for diffuse reflection. Dynamic global illumination methods compute a fast estimation of the rendering equation on the fly using various simplifications, including geometry approximations using voxels [50, 51], surfels [52, 53], or spheres [54, 55]; lighting approximations using photons [56, 57], virtual point lights [32, 58], or spherical functions [59, 60, 61]; screen space techniques [62, 63, 64, 65, 66]; caching [67, 68]; or reconstruction from sparse samples [69, 70, 71]. Our many-light solutions in Chapter 3 and 4 can provide real-time illuminations from large number of VPLs, which are effective in rendering diffuse indirect illumination.

With real-time ray tracing, the flexibility and accuracy of real-time global illumination can be dramatically improved. Dynamic diffuse global illumination [72] (or RTXGI) uses ray tracing to update light probes in real-time to achieve dynamic diffuse GI at a low ray budget. For higher quality rendering that does not rely on interpolation, each pixel needs to use an unbiased estimator of the path integral. This can be made efficient through path reuse, which is introduced in Section 2.4. We introduce methods based on path reuse in Chapter 5 and 6 that compute unbiased global illumination with general material types.

2.3 Volume Rendering

Participating media like smoke, fog, cloud, and fire are ubiquitous in real life. The rendering of heterogeneous participating media is yet another challenging topic for real-time rendering, especially when considering multiple scattering and light interactions between surfaces and particles. The volume rendering equation represents incident radiance L at point $\mathbf{x}_0$ from direction ω_o and integrates the outgoing radiance through volumetric media L^m and the surface or light behind it, L^s

$$L(\mathbf{x}_0, \boldsymbol{\omega}_o) = \int_0^{z_s} T(\mathbf{x}_0 \leftrightarrow \mathbf{x})\, \sigma_t(\mathbf{x})\, L^m(\mathbf{x}, \boldsymbol{\omega}_o)\, dz \\ + T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_s)\, L^s(\mathbf{x}_s \rightarrow \mathbf{x}_0)\ , \tag{2.1}$$

where $\mathbf{x} = \mathbf{x}_0 - z\boldsymbol{\omega}_o$ is a point along direction $\boldsymbol{\omega}_o$ towards $\mathbf{x}_0$, $\sigma_t(\mathbf{x})$ is the extinction coefficient at $\mathbf{x}$, and the transmittance function $T(\mathbf{x}_0 \leftrightarrow \mathbf{x})$ represents visibility between $\mathbf{x}_0$ and $\mathbf{x}$

$$T(\mathbf{x}_0, \mathbf{x}) = e^{-\int_0^z \sigma_t(\mathbf{x}_0 - y\boldsymbol{\omega}_o) dy}, \tag{2.2}$$

and $\mathbf{x}_s = \mathbf{x}_0 - z_s\boldsymbol{\omega}_o$ is the corresponding surface along the ray. L^m includes volumetric emission L^m_e and in-scattering

$$L^m(\mathbf{x}, \boldsymbol{\omega}_o) = \frac{\sigma_a(\mathbf{x})}{\sigma_t(\mathbf{x})} L^m_e(\mathbf{x}, \boldsymbol{\omega}_o) + \frac{\sigma_s(\mathbf{x})}{\sigma_t(\mathbf{x})} \int_S \rho(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o) L(\mathbf{x}, \boldsymbol{\omega}) d\boldsymbol{\omega}, \tag{2.3}$$

where σ_a and σ_s are absorption and scattering coefficients with $\sigma_t(\mathbf{x}) = \sigma_a(\mathbf{x}) + \sigma_s(\mathbf{x})$, S is the sphere of all directions, and $\rho(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o)$ is the media's phase function.

With multiple scattering, computing $L(\mathbf{x}, \boldsymbol{\omega})$ inside the integral from Equation 2.3 via Equation 2.1 is costly. Real-time rendering often uses coarse approximation for multiple scattering [73]. For simple single scattering, with no volumetric emission and a few point or directional lights, volumetric shadow mapping can achieve real-time rendering performance [74, 75, 76, 77, 78, 79]. But these are inefficient for more general lighting conditions and multiple scattering. Modern games use "froxel" representations [80] to align volumes with the view frustum to minimize memory incoherence during traversal, but otherwise behave similarly.

2.3.1 Monte Carlo Sampling for Volume Rendering

Path tracing and Monte Carlo sampling provide a more general solution for integrating the volume rendering Equations 2.1, 2.2, and 2.3.

Importance sampling distance z (Equation 2.1) in a homogeneous volume (constant σ_t) with a probability distribution function (PDF) proportional to $\sigma_t T(\mathbf{x}_0, \mathbf{x})$ is trivial; the resulting cumulative distribution function (CDF) $1 - e^{-z\sigma_t}$ is easily inverted. The resulting z values are called the *free-flight distance*. In heterogeneous volumes, *regular tracking* [81] represents $\sigma_t(\mathbf{x})$ with piecewise simple functions with analytically invertible CDFs, allowing tracking of each piece separately to find scattering events via an exact PDF. In general heterogeneous volumes, *ray marching* finds these events with an approximated PDF

[82], which introduces bias to the result. *Null-collision* methods avoid bias by introducing fictitious media to simplify the CDF. For example, delta tracking [83, 84] uses piecewise constant majorant $\bar{\sigma}$ (for $\bar{\sigma} \geq \sigma_t$) and determines null collisions via a secondary Monte Carlo process, but the sampling PDF is generally not available in closed form.

Recently, Miller et al. [9] introduces a special path space formulation including null scattering to obtain analytical PDFs. But null-collisions can reduce performance in highly uneven volumetric densities, when long chains of short null collisions occur. Acceleration structures (e.g., super-voxels [85] and kd-trees [86]) partition space with separate, tighter majorants to improve performance. Decomposition tracking [5] reduces overhead by splitting media into a constant density control volume and a residual volume, where tracking occur separately and the minimum distance is taken. Kutz et al. [5] introduces weighted delta tracking [87] to allow non-tight upper bounds in the residual volume.

Distance sampling techniques also apply to transmittance estimation. For example, delta tracking gives a per-sample binary transmittance decision, based on if a real collision occurs. *Ratio tracking* modifies delta tracking, replacing stochastic termination with its expectation, giving non-binary transmittance and improving convergence speed. *Residual ratio tracking* uses fewer steps for ratio tracking by separating residual and control components of the extinction function [6] and applies in various contexts [88, 89]. Recent next-flight estimators [82] improve delta and ratio tracking efficiency if the fictitious media has a lower density.

New integral formulations using power-series expansion improve transmittance estimation via sample stratification [90]. Kettunen et al. [91] propose unbiased ray marching that corrects biased methods with occasional higher order terms, leveraging ray marching efficiency to compute low-noise transmittance.

Importance sampling $\boldsymbol{\omega}$ in the phase function ρ [92] is similar to any bidirectional scattering distribution function. However, explicitly sampling light sources with *next event estimation* (NEE) often increases efficiency. NEE can be combined with other sampling via *multiple importance sampling* (MIS). Miller et al. [9] use MIS to integrate in path space using previously unknown PDFs. To efficiently integrate volumetric emission, Simon et al. [93] introduce forward next event estimation (FNEE) that samples solid angle to perform line integration.

All these methods consider local sampling, rather than importance sampling full paths. This limits quality, increasing samples needed to converge and reducing their real-time appeal. Recent denoisers can filter noisy samples to a final image. But sample counts must be sufficient to achieve post-filter temporal stability. Thus, improving sample quality is vital, even with state-of-the-art denoising.

2.3.2 Bidirectional Volume Rendering

Bidirectional methods consider entire paths to improve sampling quality. Equiangular sampling [94] jointly samples a light vertex and the penultimate path vertex receiving in-scattered light. Joint importance sampling [95] allows double scattering with a joint distribution. Zero-variance random walks further extend joint sampling, building random walks with near-zero variance by considering all terms in the path integral. In some scenarios, this applies to subsurface scattering [96, 97] and can improve path guiding [98] in general participating media.

Other bidirectional techniques estimate photon density using volumetric photon maps [99]. Density estimation queries can be improved using beams [100]. Higher dimensional primitives, such as photon beams [101], photon planes and volumes [102], and photon surfaces [103] can significantly reduce variance.

Bidirectional techniques based on virtual lights [104, 105] also benefit from higher dimensional light representations. Combining different kinds of density estimation with path tracing [106] often provides a robust framework to optimally sample across various scenes.

However, bidirectional methods often use complex data structures with costly generation and maintenance phases. This adds often insurmountable engineering complexity in real-time contexts.

2.4 Path Reuse

To enable higher quality and more general global illumination effects that potentially contain high-frequency information, popular real-time GI methods that rely on subsampling, scene simplification, and interpolation are insufficient. Using path tracing to generate independent samples on each pixel can avoid blurring the signal in principle, but it remains impractical due to an insufficient number of samples that can be generated in real-time. A potential solution is path reuse [11]. The principle is already widely applied in photon mapping and VPL rendering. But path reuse between pixels avoids the complexity of density estimation and the strong correlation of using the same set of light paths for all pixels. Path reuse can dramatically reduce variance in equal time compared to path tracing with independent samples in diffuse scenes, though it imposes fairly high storage and computation costs; paths must explicitly remain in memory to benefit others, and visibility rays are required to connect neighbors.

More recent work reuses paths via finite differences in the gradient domain [107], which can also apply to path tracing [108] and volume rendering [109].

Spatiotemporal reservoir resampling (ReSTIR) [7] is a recently introduced importance sampling method which combines resampled importance sampling (RIS) [**?**] and screen-space spatiotemporal path reuse. By defining a target function that approximates the integrand, spatiotemporal resampling reuses samples indirectly to improve PDFs, rather than explicitly reusing paths for shading. In this sense, it is more akin to path guiding [110], if done in a feed-forward, streaming fashion. The ReSTIR method reuses a large number of samples spatiotemporally to dramatically improve the resulting sample distribution compared to RIS. ReSTIR was first applied to direct lighting with reuse in screen space [7], where the integration domains are fixed across pixels (i.e., the surface of all lights). Recent work extends ReSTIR to world-space sample reuse [111, 112] and longer paths [10] for global illumination, where integration domains and reasoning about correctness become more complex. Chapter 5 shows our extension of ReSTIR to volumetric path space to provide low-noise, interactive rendering of heterogeneous volumes in complex lighting environments.

However, like other screen space path reuse techniques, ReSTIR leverages correlations between pixels to achieve efficiency. With high geometric complexity or narrow BSDF profiles, nearby pixels can have low correlation in their integrands. Reusing such neighbors can often reduce the sampling quality instead of improving it. How to robustly reuse spatiotemporal neighbors is still an open question. In Chapter 6, we introduce a generalization of the resampled importance sampling theory to allow defining advanced shift mappings between sampling domains to maximize the effectiveness of path reuse. In

Section 2.4.1 and Section 2.4.2, we introduce related work on shift mappings and briefly explain some key resampling algorithms as they are the foundation of the path reuse algorithms in this dissertation.

2.4.1 Shift Mappings

Path reuse algorithms evaluate pixel color by reusing path samples between pixels. As in ReSTIR and RIS, the formulation of Bekaert et al. [11] requires path samples to come from a shared domain, while more recent work [21] allows different integration domains and explicitly defines *shift mappings* to map paths between them. This better reuses complex light transport paths, including specularity from glass and mirrors. By extending RIS and ReSTIR to utilize general shift mappings, our method in Chapter 6 handles these complexities better than prior work that uses ReSTIR for GI [10].

Shift mappings originally arose in gradient-domain rendering [107, 108], where the image is reconstructed with discrete image gradients, evaluated by subtracting a path's contribution from its copies *shifted* into adjacent pixels.

Many shift mappings have been developed: reconnecting to the earliest rough vertex [107], manifold exploration shifts [107] and half-vector copying [108] for specular transport, random number replay [113, 108, 114], and numerous extensions to e.g., bidirectional path tracing [115], photon mapping [116, 109], participating media [109], vertex connection and merging [117], and spectral rendering [118]. The recent survey of Hua et al. [114] provides a more in-depth view to shift mappings and gradient-domain rendering.

2.4.2 Resampling Algorithms

Here we provide a brief review of resampling algorithms related to methods proposed in this dissertation. More detailed introductions will be provided in Chapter 5 and 6.

2.4.2.1 Sampling Importance Sampling

Recent resampling methods build on *sampling importance resampling* (SIR) [119]. SIR obtains better-distributed samples $(Y_i)_{i=1}^{N} = (Y_1, \dots, Y_N)$ by subsampling a set of independent and identically distributed (i.i.d.) samples $(X_i)_{i=1}^{M}$ proportional to resampling weights $w_i = \hat{p}(X_i)/p(X_i)$, where $\hat{p}(x)$ represents a desired (potentially unnormalized) target distribution. As M grows, the distribution of samples Y_i converges to $\bar{p} = \hat{p}(x)/\|\hat{p}\|_1$. See

Guetz [120] for an in-depth overview of SIR.

2.4.2.2 Resampled Importance Sampling (RIS)

Resampled Importance Sampling [8] introduces RIS, which provides proper normalization for SIR-selected samples when used in Monte Carlo integration. RIS extends SIR to allow sourcing $(X_i)_{i=1}^{M}$ from different probability distributions in a single domain and provides multiple importance sampling (MIS) weights that ensure convergence to the target distribution in such cases.

2.4.2.3 Reservoir-Based Resampling

Chao [121] introduces a reservoir sampling algorithm that selects a random sample from the input set $(X_i)_{i=1}^{M}$ in a single-pass streaming manner. A *reservoir* stores the selected sample, current stream length M, and sum of weights w_i $(i \leq M)$; each new stream element X_i replaces the selected sample with probability $w_i / \sum_{j \leq i} w_j$. Reservoir sampling pairs naturally with resampling; combined, they perform RIS in a streaming manner with a constant memory footprint. ReSTIR [7] uses chained reservoir resampling to share samples across pixels and frames. It alternately generates new independent samples for each reservoir (e.g., per pixel) and reuses samples between similar reservoirs (i.e., domains). Sharing well-distributed samples across integration domains improves sample distribution and amortizes costs of generating initial samples.

CHAPTER 3

REAL-TIME RENDERING WITH LIGHTING GRID HIERARCHY[1]

In this chapter, we present an extension of the lighting grid hierarchy method for real-time rendering with many lights on the GPU. We describe efficient methods for parallel construction of the lighting grid hierarchy and using it with deferred rendering. We also present a method for estimating shadows from many lights with a small number of shadow samples using the ray tracing API on the GPU. We show how our approach can be used for real-time global illumination computation with virtual point lights.

3.1 Introduction

Rendering with a large number of light sources (i.e., the many-lights problem) has been an important challenge in computer graphics. While there exist elegant offline rendering methods that provide sublinear performance in the number of light sources [26, 43, 16], it still remains an open problem for real-time rendering.

In this chapter, we provide an extension on the recently-introduced lighting grid hierarchy method [16], which was originally developed for rendering explosions by representing their illumination using many point lights, and we make it suitable for general-purpose real-time rendering on the GPU. Given a large number of light sources, we construct a lighting grid hierarchy on the GPU and use it for efficiently approximating the total lighting contribution from all lights in a deferred renderer. We achieve this by rendering the lights within the lighting grid hierarchy as range-limited light volumes and using a small number of shadow samples for approximating the shadows from all lights via a new importance sampling algorithm. The computation of the chosen shadow samples is performed using the recently-introduced ray tracing API on the GPU and a screen-space

[1]D. Lin and C. Yuksel, "Real-time rendering with lighting grid hierarchy," Proceedings of the ACM in Computer Graphics and Interactive Techniques (Proceedings of I3D 2019), vol. 2, no. 1, pp. 8:1–8:17, 2019.

filter is used for eliminating the high-frequency noise of shadow sampling. We show how our method can be used for computing global illumination (Figure 3.1) with a large number of virtual point lights at real-time frame rates. The technical contributions in this chapter include:

- An efficient method for lighting grid hierarchy construction on the GPU,
- An importance sampling algorithm for estimating shadow contributions of all lights with a fixed memory footprint,
- A hybrid ray tracing-rasterization approach for high-quality diffuse-dominant global illumination in complex scenes using many virtual lights, including dynamic lighting and dynamic geometry at real-time frame rates.

3.2 Background: VPL and Lighting Grid Hierarchy

In this section we briefly overview the related work in computer graphics regarding VPL rendering. We also provide a summary of the lighting grid hierarchy method [16].

3.2.1 Prior Work in VPL Rendering

Most interactive global illumination methods aim to provide a fast estimation of the rendering equation [122]. The method we describe in this chapter uses virtual point lights (VPLs) [32]. Advantages of using VPLs include easy implementation, stable appearance,

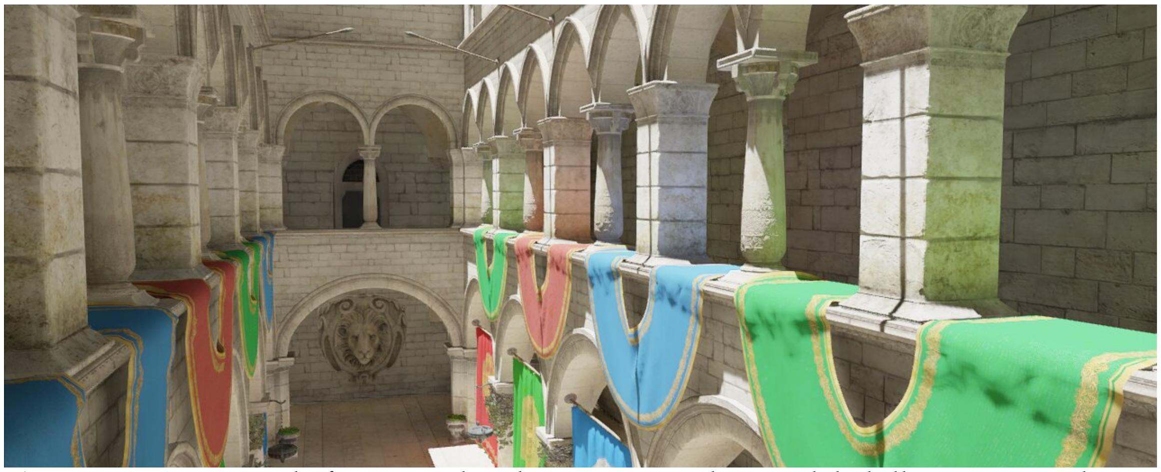


Figure 3.1: An example frame rendered using our real-time global illumination solution with one million virtual point lights, computed by our method, using $\alpha = 2$ and 4×4 interleaved sampling. The render time is 24 ms on an NVIDIA RTX 2080 GPU at 1280×720 resolution.

and scalability. Due to the low-frequency nature of diffuse reflection, VPLs are particularly effective in rendering diffuse indirect reflection, which, in many cases, is the most important part of global illumination. However, the singularity of point lights causes practical problems. Simply clamping the inverse square attenuation leads to energy loss. Energy compensation methods that use path tracing [123], screen space sampling [124] or a mixture with photon mapping [125] have been developed to solve this issue, but they are computationally expensive, particularly for real-time rendering.

An important obstacle for using VPLs in real-time rendering has been the challenge of efficiently handling many light sources. Clustered shading [126] is the first method that presented real-time rendering performance with one million point lights; however, it assumes local illumination. Stochastic light culling [127] achieves interactive rates by fitting VPLs into the tiled shading framework [128], but introduces banding artifacts that are difficult to filter. Forward light cuts [129] can compute the illumination of many VPLs using a multi-scale radiance cache, but shadows are not accounted. More recently, Estevez and Kulla [41] introduced an efficient method for importance sampling many lights during path tracing by stochastically traversing a bounding volume hierarchy of light clusters, and this method is recently extended to real-time rendering [130].

Computing shadows from many VPLs has been another related challenge. There are solutions for real-time shadow computation from hundreds of lights [131, 132], but scenes using VPLs for indirect illumination usually require thousands of VPLs or more. Traditionally, shadows are computed for each of the VPLs, but this can be too expensive for real-time rendering, unless combined with a subsampling technique. Harada et al. [133] proposed a method for efficiently casting shadow rays to lights within each render tile, but it does not solve the problem for virtual point lights with potentially global influence radius. Imperfect shadow maps [134] use a point cloud representation of the scene geometry to significantly reduce the shadow mapping cost at the expense of shadow quality.

Recently, the lighting grid hierarchy method [16] was introduced for rendering explosions by representing their illumination using many virtual point lights. We extend this method in this chapter by providing a GPU-friendly variant that is suitable for real-time rendering with many lights, so we discuss this method in more detail below (Section 3.2.2).

3.2.2 Lighting Grid Hierarchy

The lighting grid hierarchy method [16] provides an effective solution to the many-lights problem, though it was originally introduced for rendering explosions with self-illumination by representing the volumetric illumination data as many virtual point lights. As opposed to alternative solutions to the many-lights problem, lighting grid hierarchy provides a temporally stable computation. It also allows efficiently precomputing and storing shadows for all lights, which leads to orders of magnitude faster computation with volumetric shadows needed for rendering explosions. In this chapter we extend this approach by providing an efficient parallel construction method, presenting a technique for efficiently computing the lighting from the hierarchy on the GPU, and introducing an importance sampling algorithm that avoids shadow precomputation, all of which are crucial for achieving real-time frame rates.

The lighting grid hierarchy method represents the entire illumination from all lights at multiple resolutions. Each level of the hierarchy corresponds to a different resolution representation that approximates the original lights using fewer light sources. A level is constructed by placing a volumetric grid that encapsulates the original lights. The vertices of the grid approximate the lights around them, such that the contribution of each original light is distributed to the eight neighboring grid vertices using trilinear weights. A *grid light* is generated from each grid vertex with non-zero illumination and placed at the illumination center of the original lights it represents. The highest resolution grid forms level 1 with the set of light sources $\mathbb{S}_1$. Higher levels ℓ of the hierarchy use grid cells with twice the size in all dimensions as compared to the level $\ell - 1$ right below them. The highest (coarsest) level $\ell_{\max}$ typically has a single cell with 8 vertices, forming $\mathbb{S}_{\ell_{\max}}$. Therefore, the number of levels constructed depends on the resolution of level 1. The original lights are kept at level zero, forming $\mathbb{S}_0$.

For providing an efficient solution to the many-lights problem, a lighting grid hierarchy approximates the light coming from different distances using different levels of the hierarchy, providing different resolution representation of the original lighting. This is accomplished using *blending functions* that form a partition of unity for any distance from the point where lighting is computed (Figure 3.2). These blending functions determine the influence regions of the lights at each grid level and the incoming illumination from a

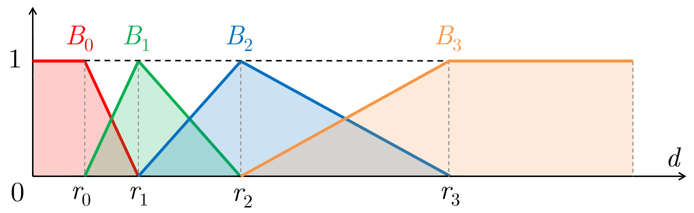


Figure 3.2: The blending functions B_0, B_1, B_2, and B_3 of lighting grid hierarchy with $\ell_{\max} = 3$, forming a partition of unity for any distance d from the point where lighting is computed.

light is modulated by the corresponding blending function value. Let h_ℓ be the grid size of level ℓ. The non-zero regions of the blending functions are determined by distances $r_\ell = \alpha h_\ell$, where α is a user-defined parameter that determines the accuracy of the lighting approximation. Larger α values lead to blending functions with larger non-zero regions and result in using more grid lights for estimating lighting with higher accuracy.

3.3 Rendering with Many Lights

Our rendering algorithm uses the lighting grid hierarchy method [16] to efficiently evaluate the illumination from a large number of point lights. In our experiments we use this algorithm for computing indirect illumination with many virtual point lights (VPLs) [32], though our lighting computation is independent from how the point lights are generated.

We begin with constructing a lighting grid hierarchy from the given point lights on the GPU (Section 3.3.1). We use this lighting grid hierarchy within a deferred renderer for efficiently estimating the illumination from all lights (Section 3.3.2). While computing the lighting, we stochastically pick a fixed number of shadow samples to be computed via ray tracing on the GPU (Section 3.3.3). Finally, we filter the computed shadows to eliminate the high-frequency sampling noise and use the result as shadow ratio estimators for computing the final lighting approximation.

3.3.1 Lighting Grid Hierarchy Construction

The problem of constructing a lighting grid hierarchy is similar to the particle-to-grid transfer operations used in hybrid Lagrangian-Eulerian simulation systems [135, 136]. Each level of the hierarchy can be constructed using either *scatter* [136] or *gather* [135] operations. The scatter approach loops over each light and adds its illumination to the corresponding grid vertices. Since the parallel scatter loop involves atomic operations, it can be highly inefficient for higher (coarser) levels of the hierarchy, as the small number of target grid vertices at these levels lead to frequent thread contentions in atomic operations. The gather approach, on the other hand, loops over each grid vertex and finds the corresponding lights that contribute to the vertex. This eliminates the need for atomic operations, but requires search operations for finding the corresponding lights. This search can be accelerated by a pre-ordering step [135], which can also be expensive to compute.

To provide an efficient parallel construction algorithm, we split the construction process into two steps. In the first step we scatter the contributions of each input light to the first grid level $\mathbb{S}_1$ with the highest resolution. Since this level involves a relatively small percentage of thread contentions, the related atomic operations can be performed efficiently. In the second step we build the rest of the levels using a gather approach. To avoid an expensive pre-ordering step, we build these levels using the grid lights of the first level $\mathbb{S}_1$, which are already ordered by construction.

This approach, as opposed to generating all levels directly from the input lights $\mathbb{S}_0$, leads to some smoothing in the final lighting approximation from the lighting grid hierarchy, but provides a highly efficient mechanism for the parallel construction process. Since VPLs are placed only on surfaces, a significant portion of the grid vertices in the volume may contain no illumination, especially for the lower (finer) levels of the hierarchy. Therefore, the construction process is completed by a stream compaction pass that is applied for each level to remove the large percentage of unused grid vertices.

Since we use a bottom-up construction of the hierarchy by building $\mathbb{S}_1$, we must first determine the size of the grid cells h_1. We begin with computing the bounding box of all input lights and set the grid size of the top level $\mathbb{S}_{\ell_{\max}}$, which only contains a single cell (i.e., 8 grid lights), as the longest edge of this bounding box. The grid size for the first level $\mathbb{S}_1$ is computed using $h_1 = h_{\ell_{\max}} / 2^{\ell_{\max}-1}$.

In our implementation the number of lighting grid hierarchy levels $\ell_{\max}$ is controlled by a user-specified parameter.

3.3.2 Lighting Computation

If the lighting grid is densely populated, such that each grid vertex contains a light with non-zero intensity, lights around a shaded point can be directly gathered from the grid. However, the stream compaction pass we use for eliminating grid lights with zero intensities prevents trivially finding the lights around shaded points from their grid locations. Therefore, we use the lighting grid hierarchy within a deferred renderer for estimating the illumination from all lights with a light rasterization step. After generating G-buffers for the scene geometry, we rasterize the lights in the lighting grid hierarchy as (coarse) spheres (approximated using cubes in practice) [137]. Since the blending function values for the lights in the lighting grid hierarchy are zero beyond the distance $2r_\ell$ from the light sources, for each light we draw a sphere with $2r_\ell$ radius. The exception is the 8 lights in the top level of the hierarchy, which use blending functions that do not go to zero with increasing distance, so these lights can be handled separately by drawing a screen-size quad. This process produces fragments for each pixel that the lights can potentially illuminate with a non-zero blending function value. Yet, the blending functions can still evaluate to zero for some of these fragments, since each grid light at the higher levels of the hierarchy has a minimum illumination radius $r_\ell/2$ within which the blending function is zero (see Figure 3.2). Therefore, we compute the blending function for each fragment and discard the fragment if it is zero.

We perform the lighting computation for each fragment with a non-zero blending function value and accumulate the result without considering shadows. Shadows are computed separately, as explained below (Section 3.3.3).

3.3.3 Shadow Sampling

The lighting grid hierarchy allows estimating the illumination using a small subset of all lights. Yet, in practice lighting computation of each pixel still involves hundreds of lights with non-zero blending function values (especially with relatively large α parameters) and computing shadows for each one of these lights can be prohibitively expensive for real-time rendering. Even though the lighting grid hierarchy method allows precomputing shadows (such as shadow maps) with reasonable memory usage, this precomputation can easily become the bottleneck of the entire rendering process. Therefore, we instead stochastically estimate shadows using a fixed number of samples, which are computed via ray tracing on the GPU. We pick these shadow samples during the lighting computation (Section 3.3.2). The shadows computed from these samples are then used as shadow ratio estimators [22].

We must pick the shadow samples randomly, independent of the order in which the lights are processed, to avoid introducing bias in shadow sampling. Furthermore, this random sampling should be performed independently for each pixel to avoid correlation in sampling. This process requires considering the set of all lights that have non-zero illumination contributions for each pixel. Moreover, since the illumination contributions of each light in the lighting grid hierarchy can vary drastically, using an importance sampling scheme is crucial for reducing the variance in sampling.

We pick a small number of shadow samples for each pixel during the lighting computation, while rendering the lights as spheres (Section 3.3.2). These shadow samples are evaluated after the lighting computation via tracing shadow rays on the GPU from the pixel positions towards the selected shadow sample locations. The lighting grid hierarchy we construct contains the variance of the illumination center for each light. We use these variance values for randomly picking shadow sample positions to produce area shadows, as opposed to using the illumination centers directly. Each one of the shadow samples of a pixel is selected independently. Therefore, it is possible to have multiple shadow samples of a pixel belonging to the same light source, though it is improbable in practice, considering that each pixel is illuminated by hundreds of lights. Nonetheless, even if two shadow samples of a pixel use the same light, they are likely to send shadow rays towards slightly different directions.

Let f_i be the probability density of picking the light source i for shadow sampling, such that the probability of picking the light source is $p_i = f_i / \sum_j^n f_j$, where n is the total number of lights in the hierarchy. These f_i values are determined per pixel based on the illumination contribution of each light for importance sampling, such that f_i is non-zero if and only if the light has non-zero illumination contribution (disregarding potential shadowing). We use weighted reservoir sampling[121] to pick light samples with constant space. During lighting computation, we store a running total for the cumulative probability density $\hat{f}_i = \sum_j^i f_j$ for each pixel. For each fragment with a non-zero f_i value, we decide whether to use it as a shadow sample stochastically with probability $\tilde{p}_i = f_i / \hat{f}_i$, using the accumulated probability density $\hat{f}_i$ while rendering the light source i. This stochastic decision is performed separately for each shadow sample of the pixel. If the light is selected as a shadow sample, it overwrites the previously selected sample. Note that the first light of a pixel with $\tilde{p}_1 = f_1 / \hat{f}_1 = 1$ is always selected as a shadow sample, though succeeding lights can overwrite the shadow sample. At the end of the lighting computation, this process provides k shadow samples each with the desired probability p_i, where k is the number of shadow samples per pixel, controlled as a user-defined parameter.

After the lighting computation, during which k shadow samples are picked, we trace shadow rays on the GPU to determine a binary shadow value for each sample. The average of the k samples provides the shadow value for the pixel, which can be used as a shadow ratio estimator [22].

Stochastic shadow sampling, as explained above, leads to a substantial amount of noise when using a small number of shadow samples. For eliminating the high-frequency noise in shadow sampling a screen-space bilateral filter can be applied to the computed shadow values before using them as shadow ratio estimators [22]. In our tests we use a wavelet-based filter [138], which we found to be more effective for filtering the shadow noise of VPLs used for computing diffuse-dominant global illumination.

3.4 Implementation and Results

We evaluate our method by computing indirect illumination with VPLs [32]. The VPLs are generated by tracing light rays up to three diffuse bounces. When the lighting condition changes, we regenerate all VPLs and construct a new lighting grid hierarchy. We use the DirectX ray tracing API for both generating VPLs and computing shadow rays. All performance results are generated using an NVIDIA RTX 2080 graphics card at 1280×720 resolution.

3.4.1 Additional Optimizations

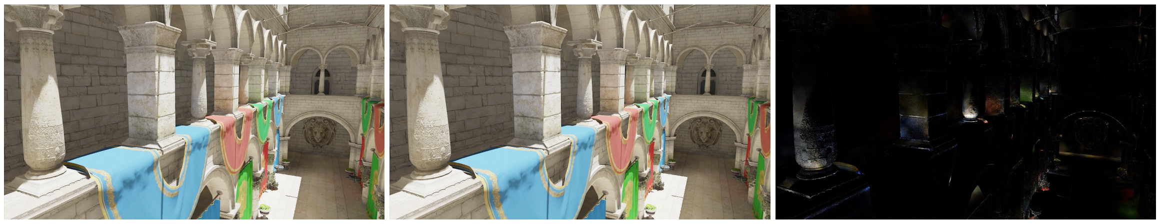
The process of picking the shadow samples (Section 3.3.3) involves atomic operations for updating the running total for the cumulative probability density $\hat{f}_i$ and overwriting the shadow samples. However, in practice the impact of thread contentions during the lighting computation on the final result can be negligible. An exception is rendering to very small viewports. In our tests we found that disabling thread locks produces virtually identical results with 10–20% improvement in render times. Therefore, the results in this chapter are generated without thread locks, unless otherwise indicated.

Note that each level of the lighting grid hierarchy encodes the entire illumination of all input lights with a different resolution. Therefore, it is possible to skip using the actual input lights altogether and begin the lighting computation using the first level of the hierarchy $\mathcal{S}_1$ instead. This can significantly reduce the overdraw caused by rendering spheres for each light source and accelerate the lighting computation, but it also introduces some smoothing to the estimated illumination. Yet, in the case of using VPLs for computing indirect illumination, this additional smoothing can even be preferable in practice. Therefore, all results in this chapter are generated using the lighting grid hierarchy starting from $\mathcal{S}_1$.

In addition, due to the large number of grid lights in $\mathcal{S}_1$ and their relatively small illumination radii, skipping $\mathcal{S}_1$ grid lights for shadow sampling can significantly reduce the memory bandwidth and computation time, without obvious impact on the render quality. In our tests we observed an additional 10–15% speedup by skipping $\mathcal{S}_1$ for shadow sampling with almost identical render results, as can be seen in Figure 3.3. Therefore, the results in this chapter do not use $\mathcal{S}_1$ for shadow sampling, unless otherwise specified.

3.4.2 Lighting Grid Hierarchy Construction

Rendering begins with constructing a lighting grid hierarchy, which is reconstructed every time the illumination changes and a new set of VPLs is generated. Table 3.1 compares the computation time of parallel lighting grid hierarchy construction using scatter



(a) Sample shadows from $\mathcal{S}_1$ *Render time: 32.5 ms* (b) Sample shadows from $\mathcal{S}_2$ *Render time: 28.5 ms* (c) Difference ×8

Figure 3.3: Comparison of shadow sampling with and without using the lowest (finest) level of the hierarchy $\mathcal{S}_1$, using 100K VPLs, 4 shadow samples, and $\alpha = 1$.

Table 3.1: Breakdown of the hierarchy construction time (100K VPLs in *Crytek Sponza*).

	Scatter VPLs	Gather from $\mathcal{S}_1$
Compute bounds	1.7 ms	1.7 ms
Compute $\mathcal{S}_1$	2.3 ms	2.3 ms
Compute $\mathcal{S}_2$	2.3 ms	1.0 ms
Compute $\mathcal{S}_3$	4.7 ms	1.0 ms
Compute $\mathcal{S}_4$	23 ms	2.0 ms
Compute $\mathcal{S}_5$	107 ms	1.0 ms
Compute $\mathcal{S}_6$	405 ms	1.1 ms
Compute $\mathcal{S}_7$	1,563 ms	1.5 ms
Merge levels	0.5 ms	0.5 ms
Total	**2,110 ms**	**12.1 ms**

operations on the input VPLs and our gather approach using $\mathcal{S}_1$ for computing the higher levels of the hierarchy. The computation time of each step is listed in the table. The first step computes the collective bounding box of the lights and the final step merges the grid lights of all levels into a single buffer to avoid multiple draw calls during light rasterization. Notice that our gather method is more than an order of magnitude faster than the scatter approach. The first two (compute bounds and compute $\mathcal{S}_1$) and the last (merge levels) operations are identical in both cases. The difference in performance comes from the thread contentions of the scatter operations for computing the higher levels of the hierarchy. In comparison, we can efficiently construct a lighting grid hierarchy by generating higher levels from $\mathcal{S}_1$.

A qualitative comparison of lighting grid hierarchy construction methods is provided in Figure 3.4. Notice that the two methods for parallel lighting grid hierarchy construction produces similar results. Yet, an extra level of smoothing and light leakage can be observed when using our gather operations from $\mathcal{S}_1$ (Figure 3.4b).

As one would expect, the lighting grid hierarchy construction time depends on the number of VPLs. The total construction times for different numbers of VPLs can be seen in Table 3.2, showing that the construction time using our gather approach grows sublinearly with the increasing number of VPLs.

3.4.3 Rendering

Figure 3.1 shows an example image rendered using our method for computing global illumination with VPLs. As expected, our method can produce high-quality global illu-

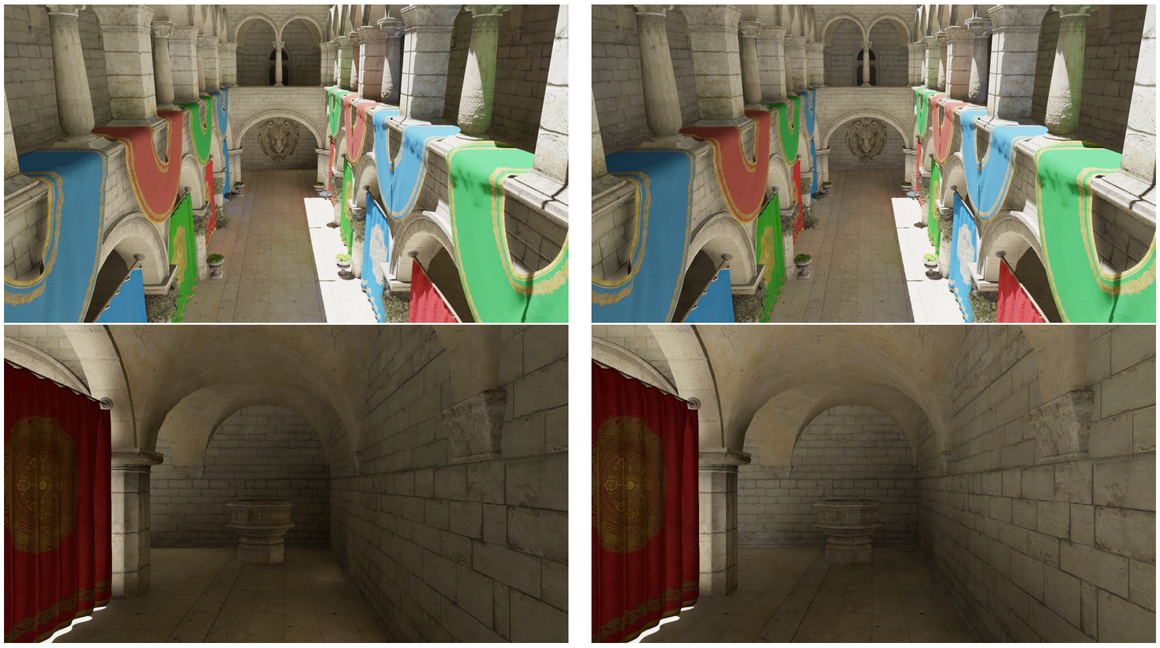

(a) Scatter VPLs (b) Gather from $\mathcal{S}_1$

Figure 3.4: Lighting grid hierarchy construction methods using (a) scatter operations on the input VPLs and (b) our gather operations using the first level $\mathcal{S}_1$ of the hierarchy, producing similar results.

Table 3.2: Computation and render times with different numbers of VPLs. The timings are generated using the camera angle in Figure 3.4.

Number of VPLs	**1K**	**10K**	**100K**	**1M**
VPL Generation	0.1 ms	0.2 ms	0.4 ms	2.5 ms
Hierarchy Construction	7.0 ms	9.2 ms	12.1 ms	32.0 ms
Render time (no shadow)	5.6 ms	8.1 ms	11.2 ms	16.0 ms
Render time (1 shadow/pixel)	10.1 ms	14.3 ms	18.4 ms	24.3 ms
Render time (4 shadows/pixel)	15.5 ms	22.4 ms	28.5 ms	37.0 ms

mination, since we can efficiently compute lighting from a large number of VPLs. The performance and the quality of our results depends on the number of VPLs used, the number of shadow samples per pixel, and the parameter α of the lighting grid hierarchy method that determines the number of light samples used for estimating the illumination.

Obviously, using more VPLs leads to a better approximation of global illumination and it also increases the render time. Since we do not directly use the VPLs in the lighting computation, the performance of our results depend on the number of grid lights in the higher levels of the hierarchy. In our tests we set the number of levels for the lighting grid hierarchy according to the number of VPLs and we pick the largest number of levels, such that the number of lights in S_1 is less than half of the number of VPLs. Thus, in our test the number of lights in the hierarchy is roughly proportional to the number of VPLs. However, as shown in Table 3.2, the render time does not linearly scale with the number of VPLs and we achieve a sublinear growth in render time with increasing VPL count.

Table 3.2 also shows that using a single shadow sample per pixel is significantly cheaper than 4. It is important to note that most of the extra cost of having additional shadow samples is related to the process of picking shadow samples during lighting computation. The actual shadow computation via tracing shadow rays is much faster in comparison.

The sublinear growth in render time with increasing VPL count can also be observed by investigating the number of overdraws (i.e., the number of fragments generated per pixel) during the lighting computation. Note that the lighting computation is the bottleneck of our system and the actual computation is proportional to the number of fragments generated. Figure 3.5 shows heatmaps indicating the number of overdraws per pixel for different number of VPLs. Notice that the number of lights in the hierarchy grows proportional to the number of VPLs, but overdraw grows approximately logarithmically with the number of lights.

One way to reduce the number of overdraws is using interleaved sampling [139, 140, 141], which splits the frame buffer into a small number of tiles. The lights at each level of the hierarchy are distributed evenly to each tile and each light is rasterized onto only one of the tiles. Since the tiles have a much lower resolution than the combined frame buffer, lights rendered onto a tile produces fewer fragments than rendering onto the entire frame. The final image of the combined frame buffer is constructed during the final compositing pass. By reducing overdraw in lighting computation, interleaved sampling significantly improves the total render time, but also leads to additional smoothing in the final illumination estimation.

Figure 3.6 shows a comparison of images generated with and without interleaved sampling. Since indirect illumination is mostly smooth, the differences caused by the additional smoothing of interleaved sampling are not easy to notice. Yet, there exists some additional smoothing in indirect shadows, especially visible behind the draping cloth shown in the insets.

Table 3.3 provides the break-down of computation time for all rendering operations

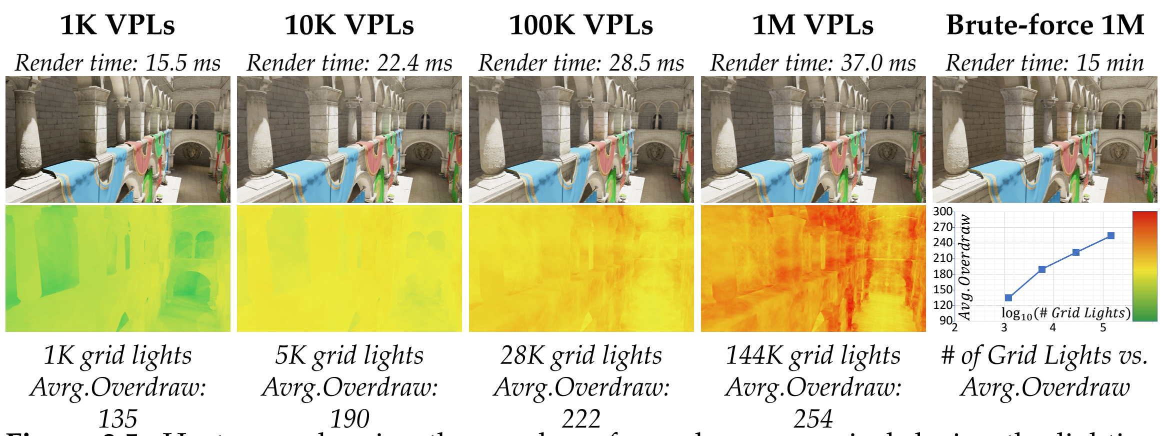
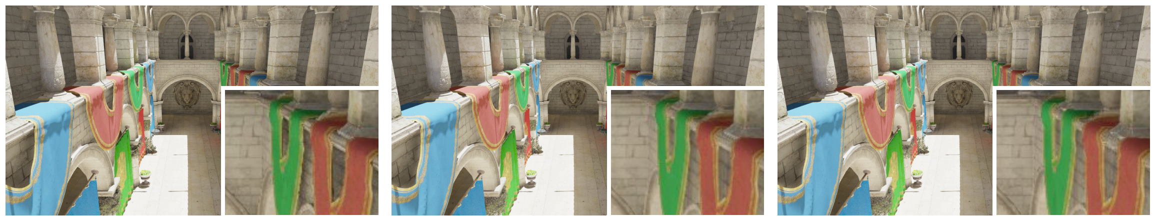


Figure 3.5: Heatmaps showing the number of overdraws per pixel during the lighting computation for lighting grid hierarchies generated from different numbers of VPLs and a log plot of number of grid lights used in lighting computation vs. average overdraw per pixel, showing the logarithmic growth in average overdraw as compared to the increasing number of lights. The first row shows the corresponding render results using our method with 4 shadow samples per pixel and the brute-force reference generated by computing shadows of each VPL.



(a) No interleaved sampling *Render time: 30.2 ms* (b) 2×2 interleaved sampling *Render time: 15.2 ms* (c) 4×4 interleaved sampling *Render time: 10.1 ms*

Figure 3.6: Comparison of our method with and without interleaved sampling using 100K VPLs, 4 shadow samples, and $\alpha = 1$.

Table 3.3: Computation time of rendering operations. The timings are generated with 100K VPLs, 4 shadow samples per pixel, and $\alpha = 1$, using the camera angle in Figure 3.6. Thread locks are used in lighting computation for only 4×4 interleaved sampling to prevent race conditions, which can be prominent in this case. The total render times do not include 0.4 ms used for generating VPLs and 12.1 ms used for constructing the lighting grid hierarchy.

Interleaved sampling	not used		2×2		4×4	
G-buffer generation	1.1 ms	3.5%	1.1 ms	7.0%	1.0 ms	5.9%
Lighting computation	24.8 ms	82.1%	9.0 ms	59.4%	3.8 ms	25.5%
Shadow ray tracing	2.3 ms	7.7%	2.2 ms	14.5%	2.2 ms	28.9%
Shadow filtering	1.9 ms	6.3%	2.8 ms	18.4%	2.9 ms	37.4%
Final compositing	0.1 ms	0.4%	0.1 ms	0.8%	0.1 ms	2.3%
Total render time	**30.2 ms**		**15.2 ms**		**10.1 ms**	

with and without interleaved sampling. Notice that without interleaved sampling lighting computation takes most of the rendering time (82.1% in this example). Using interleaved sampling with 2×2 and 4×4 significantly reduces the lighting computation time. For dynamic scenes where the lighting condition or scene geometry constantly changes, the cost of VPL generation and lighting grid hierarchy also need to be taken account into the total render cost.

Importance sampling for shadow computation is a crucial component of our method. Figures 3.7 and 3.8 compare the results of indirect shadows before and after filtering, computed with and without importance sampling. Notice that shadows are extremely noisy without importance sampling and the filtered shadows still contain a substantial amount of lower-frequency noise even with 4 shadow samples per pixel. In comparison, we achieve superior results with a single shadow sample per pixel using importance sampling. Our shadows are further improved using more samples.

The lighting grid hierarchy method with smaller α values produces a smoother lighting approximation by considering fewer lights for each lighting computation. A comparison with two different α values is shown in Figure 3.9. Since indirect illumination is mostly

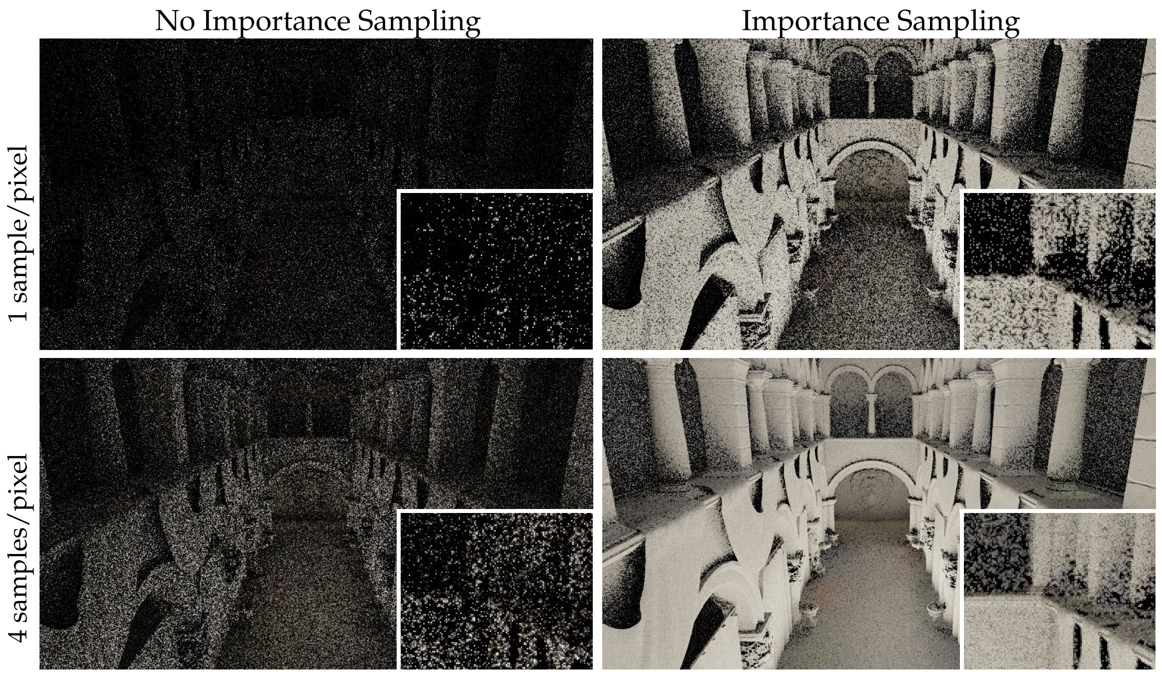


Figure 3.7: Shadowed indirect illumination without shadow filtering, generated with and without importance sampling using 1 and 4 shadow samples per pixel.

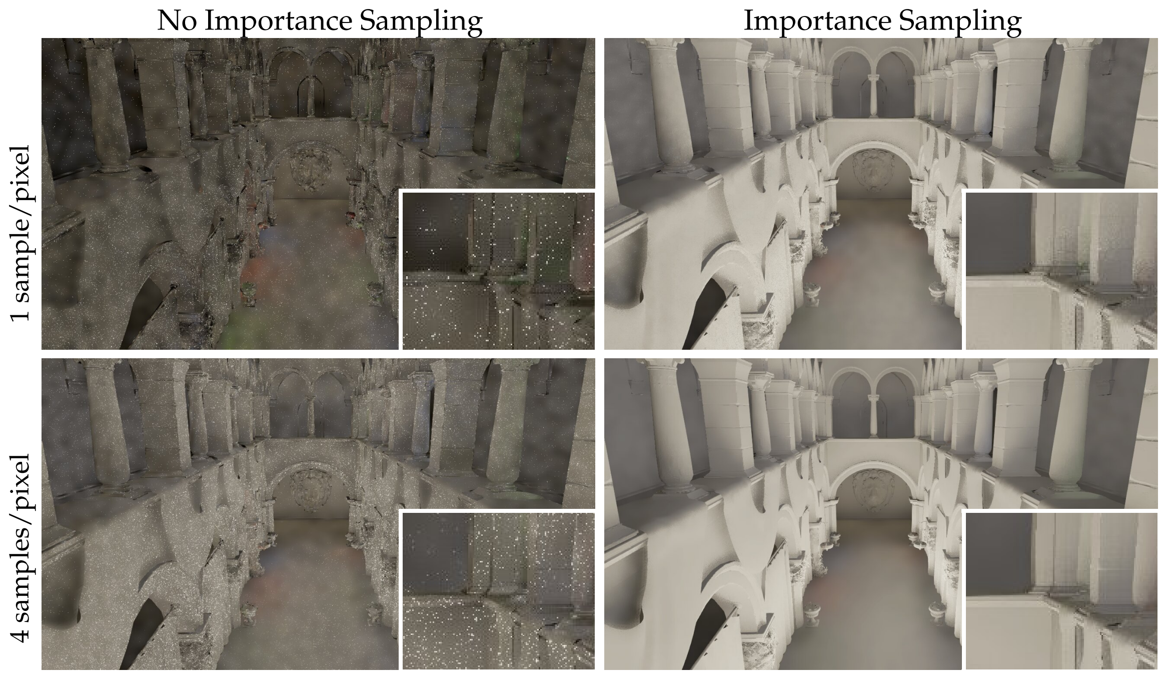
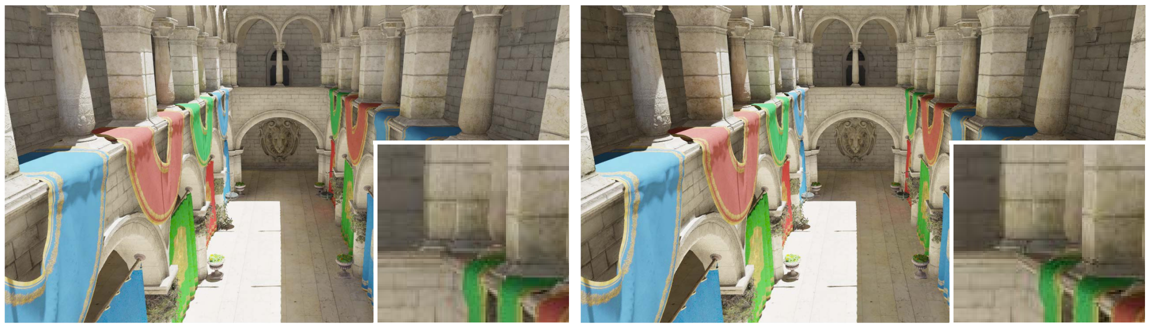


Figure 3.8: Shadowed indirect illumination with shadow filtering, generated with and without importance sampling using 1 and 4 shadow samples per pixel.

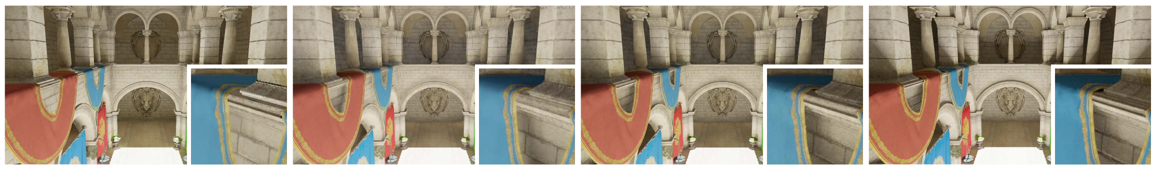


(a) $\alpha = 1$ (Render time: 31 ms) (b) $\alpha = 2$ (Render time: 93 ms)

Figure 3.9: While the accuracy of the lighting approximation improves with increasing α parameter of lighting grid hierarchy, the quality improvement in indirect illumination can be difficult to see.

smooth, the differences between the two alpha values are difficult to notice. On the other hand, doubling the alpha value triples the render time for this example.

Global illumination computation with a large number of VPLs produces high-quality results and our approach, especially when combined with interleaved sampling, provides the necessary performance for achieving real-time frame rates. We compare the results of our method to the NVIDIA VXGI implementation of voxel cone tracing [51] and path tracing [122] in Figure 3.10. Notice that using our method we can produce a reasonably close



(a) VXGI *Render time: 21 ms* (b) 4×4 interleaved samp. *Render time: 15 ms* (c) No interleaved samp. *Render time: 93 ms* (d) Path Tracing Reference *Render time: 2 hours*

Figure 3.10: Comparison of global illumination computation using (a) voxel cone tracing generated with the highest quality settings in NVIDIA VXGI 2.0, (b) our method with 4×4 interleaved sampling and $\alpha = 1.5$, (c) our method with no interleaved sampling and $\alpha = 2$, and (d) path tracing reference. The given render times for VXGI and our method do not include construction time. 100K VPLs are used for generating our results.

solution to the path tracing reference at interactive frame rates (Figure 3.10c). Using interleaved sampling with our method (Figure 3.10b) introduces additional smoothing in lighting approximation, but provides a highly efficient solution to real-time global illumination computation. Notice that the indirect shadows (e.g., behind the draping cloth and on the columns) are properly reproduced using our method, but with some extra smoothing as compared to the path tracing reference. In comparison to VXGI (Figure 3.10a), we achieve closer results to the path tracing reference with relatively less smoothing introduced to the indirect illumination estimation.

Although further reducing the render time, ignoring shadows for indirect lighting can produce unrealistic results, which is more pronounced in brightly lit scenes. Figure 3.11 shows different scenes rendered using our algorithm and compares our method with 1 and 4 shadows samples per pixel to direct illumination only and indirect illumination without shadows. Notice that without indirect shadows the resulting indirect illumination is overestimated and it becomes relatively flat.

While the lighting grid hierarchy method provides a temporally stable solution to rendering with many lights, it does not eliminate any underlying flickering of VPLs. In dynamic scenes, where the direct illumination (and/or scene geometry) changes, we regenerate all VPLs independently at each frame. Since VPL positions are distributed randomly in the scene, the resulting VPL distribution is not temporally stable, even if the scene is static. Therefore, if all VPLs are regenerated at every frame, the resulting indirect illumination estimation may include substantial amount of flickering, regardless of which method is used for computing the illumination from the VPLs. One solution is incorporating methods that

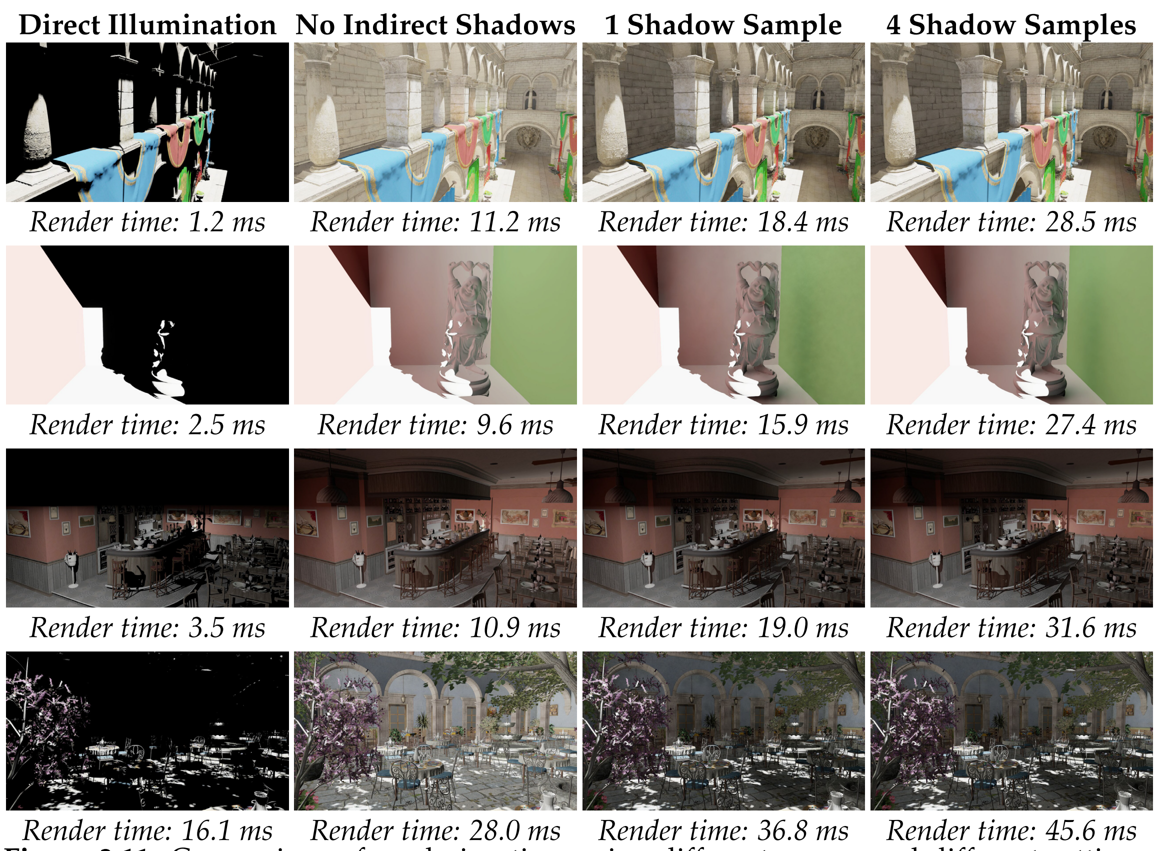


Figure 3.11: Comparison of rendering time using different scenes and different settings. All scenes are rendered using lighting grid hierarchy generated from 100k VPLs, with no interleaved sampling and $\alpha = 1$. The reported render times do not include VPL generation and lighting grid hierarchy construction. Models downloaded from Morgan McGuire's Computer Graphics Archive [1].

re-use VPLs to minimize flickering [142]. Alternatively, applying a temporal anti-aliasing (TAA) filter only to the indirect illumination can substantially reduce the flickering, but can also introduce ghosting artifacts. The solution we prefer is applying TAA only to the indirect shadows, which reduces flickering without noticeable ghosting artifacts.

3.5 Discussion and Conclusion

We have introduced an extension of the lighting grid hierarchy method that makes it suitable for real-time rendering with many lights. We have also shown how our method can be used for efficiently computing high-quality global illumination at real-time frame rates. Our method can handle dynamic scenes, including dynamic lighting.

One important issue with all efficient solutions to the many-lights problem is that light leakage is unavoidable. This is certainly the case with the lighting grid hierarchy method and our parallel construction approach that uses $\mathbb{S}_1$ for generating the higher levels of the hierarchy leads to additional light leakage.

Moreover, unless a relatively large α value is used, lighting grid hierarchy can introduce some smoothing to the illumination estimation. Our shadow computation with a small number of shadow samples introduces an additional level of smoothing due to shadow filtering. When combined with interleaved sampling, the smoothing of our lighting estimation is further amplified.

Nonetheless, our method provides an effective solution to the many-lights problem for real-time rendering.

Note that, regardless of whether VPLs are completely or partially regenerated at every frame, the lighting grid hierarchy must be reconstructed when there is any change to the VPL data. This construction introduces computation cost beyond rendering for dynamic scenes. An interesting future direction would be exploring dynamic updates to previously constructed lighting grid hierarchy that could reduce the initialization cost for dynamic scenes.

CHAPTER 4

REAL-TIME STOCHASTIC LIGHTCUTS[1]

In this chapter, we present real-time stochastic lightcuts, a real-time rendering method for scenes with many dynamic lights. Our method is the GPU extension of stochastic lightcuts [4], a state-of-art hierarchical light sampling algorithm for offline rendering. To support arbitrary dynamic scenes, we introduce an extremely fast light tree builder. To maximize the performance of light sampling on the GPU, we introduce cut sharing, a way to reuse adaptive sampling information in light trees in neighboring pixels.

4.1 Introduction

Scenes with many lights are commonplace in real-time rendering applications like video games. Yet, handling many lights has been a challenge in real-time rendering, due to the complexity of accumulating illumination from all lights for all pixels. Most game engines handle scenes with many lights using a mixture of baking and tile-based deferred rendering [128, 126] to determine which pixels should be illuminated by which lights. However, baked lighting is difficult to use with animations and tile-based deferred rendering only works if the lights have relatively small influence ranges (though stochastic ranges can be used for lights with unbounded influence ranges [127] to approximate the lighting). Recent Monte Carlo sampling methods for real-time rendering [42, 3] lift these limitations, but their computation cost limits the number of light samples that can be used at real-time frame rates, resulting in noisy lighting estimations.

We present an extension of stochastic lightcuts [4] and describe how it can be used to achieve high-performance rendering with many lights on the GPU (Figure 4.1). Our goal is to minimize the overhead of sampling from a *light tree*, so that we can afford more light samples within the same render time, achieving higher quality (i.e., lower noise). Thus,

[1]D. Lin and C. Yuksel, "Real-time stochastic lightcuts," Proceedings of the ACM in Computer Graphics and Interactive Techniques (Proceedings of I3D 2020), vol. 3, no. 1, pp. 5:1–5:18, 2020.

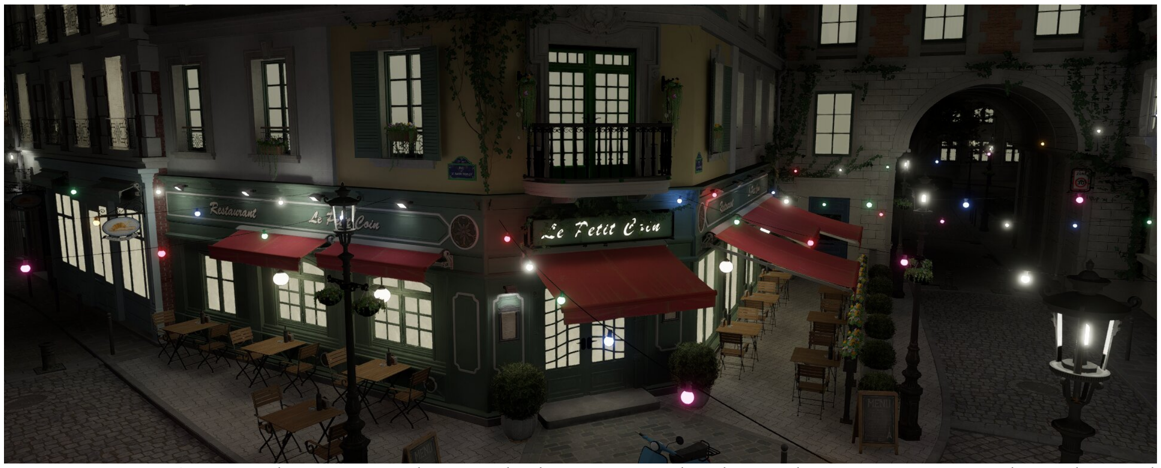


Figure 4.1: Our real-time stochastic lightcuts method renders a scene with unbiased sampling of direct lighting from 24,782 emissive triangles with a sampling time of 11.5 ms (including tracing shadow ray and lighting computation) and a total frame time of 23 ms on a NVIDIA RTX 2080 card, using 4 light samples per pixel. The screen resolution is 1920×1080. SVGF [2] and TAA are applied to filter the sampling result.

we accept sacrificing the quality of the light tree to save computation time, which can be used towards more light samples, resulting in a net gain in quality. To achieve this, we use a perfect (i.e., balanced, full, and complete) binary light tree. A perfect binary tree allows extremely fast construction and also boosts the light sampling performance. Moreover, we introduce a novel weight computation scheme for hierarchical importance sampling that provides improvements in sampling quality in our test scenes. Furthermore, we introduce *cut sharing*, which allows a block of pixels to share the same *cut* through the light tree for minimizing the overhead of cut selection. We show how cut sharing enables the use of *interleaved sampling* [141] to further improve the performance or light sampling quality. We compare our real-time stochastic lightcuts method with prior work on sampling many lights and show that our method improves the speed of light sampling, thereby providing higher quality with more light samples within the same render time budget.

4.2 Background: Stochastic Lightcuts

In this section we briefly summarize stochastic lightcuts [4]. An introduction of lightcuts [26] and many light rendering techniques can be found in Chapter 2.

Stochastic lightcuts [4] extends the lightcuts method [26]. Lightcuts builds a binary light tree prior to rendering. During rendering the light tree is evaluated from top to bottom for selecting a *cut* through the light tree. The cut is initially placed at the root node. For each node of the light tree along the cut, if the maximum possible illumination from the subtree under the node is above a threshold, the cut is moved one level below.

Stochastic lightcuts [4] uses a *hierarchical importance sampling* technique (similar to [41]) for randomly selecting a light sample within each subtree under the chosen cut, converting lightcuts to an unbiased light sampling method. For each light tree node above the cut, hierarchical importance sampling traverses the subtree below the node from top to bottom, randomly picking one of the child nodes under each node, down to a leaf node that contains a single light source, which is chosen as the light sample.

4.3 Real-Time Stochastic Lightcuts

Our real-time stochastic lightcuts method contains two key components that make it GPU friendly. First, we use a perfect binary light tree, which is extremely fast to build (Section 4.3.1) and efficient to traverse (Section 4.3.2). Second, we share cuts within $k \times k$ pixel blocks (Section 4.3.3), instead of computing a cut for each pixel during light sampling. Our cut sharing technique naturally enables interleaved sampling (Section 4.3.4), which can be used for further accelerating light sampling or improving its quality.

Our goal is to minimize the light tree construction and light sample selection times. The choices we make for achieving this goal, however, may adversely affect the quality of the tree and the light sample distribution. Yet, this reduction in sampling quality can be offset by using more light samples. Thus, our ultimate goal is using the time we save during the tree construction and light sample selection towards more light samples for a better illumination estimation.

4.3.1 Perfectly Balanced Light Tree

As mentioned above, neither the existing agglomerative or divisive light clustering methods are fast enough to fully rebuild the light tree in real-time. To solve this problem, we use a perfect (i.e., balanced, complete, and full) binary light tree. In a perfect tree, all leaf nodes that contain the individual light sources appear at the bottom level of the tree. Since the number of leaf nodes in a perfect tree must be a power of two, we add *bogus lights* as needed (Figure 4.2).

Each node in the light tree stores the bounding box of the underlying lights and the total

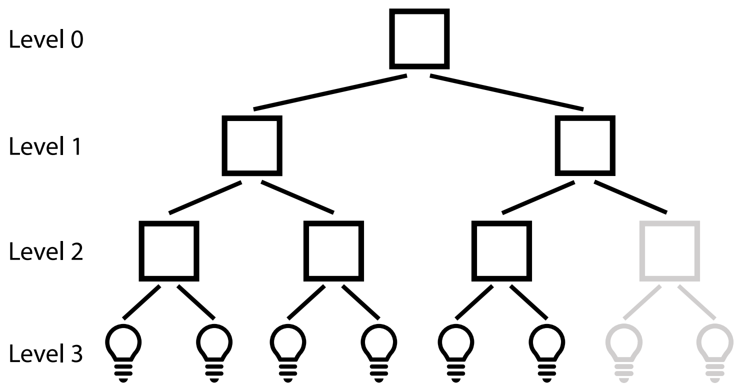


Figure 4.2: An example of a perfectly balanced light tree with four levels. Bogus lights with zero intensities are appended to the end of the leaf level (level 3) to round up the number of lights to the nearest greater power of two. Bogus lights and bogus nodes are marked with gray color.

light intensity. Since we use a perfect tree, there is no need to store child node pointers, as the child node indices can be computed directly from the parent node index. Additionally, each leaf node stores the corresponding light ID. Note that light IDs cannot be directly computed from leaf node indices in dynamic scenes, since the leaf index of a light can vary each frame.

For minimizing the construction time, our builder sorts the light sources based on the Morton code of their positions (conceptually similar to a Morton code BVH builder [143]). The leaf nodes are generated by directly copying the light information in the sorted order into the leaf nodes of the tree. Afterwards, computing the internal node data is a straightforward gathering process in a bottom-up order. For each internal node, the bounding boxes and intensities of the child nodes are simply added together. Bogus lights are assigned zero intensities and excluded from the bounding box computation. Since this gathering process is fully deterministic and commutative, we can generate level ℓ from any level ℓ_s below it, such that $\ell_s > \ell$, not necessarily the level immediately below it (where $\ell_s = \ell + 1$). This means that any combination of levels can be generated in parallel, and the construction process can be parallelized in a way that fully exploits the capability of the GPU.

While our perfectly balanced light tree is fast enough to be rebuilt for every frame, we also provide an option to use a two-level light tree, where a light tree is split into one *top-level* and multiple *bottom-levels*. This is to improve the quality of light sampling for scenes with sparsely distributed light meshes of heterogeneous sizes, where Morton code sorting on all lights might not reproduce the spatial proximity very well. In a two-level light tree, each bottom-level is built separately as a perfect binary light tree. After all bottom-levels are built, they are used as leaf nodes of the top-level tree, which is also built as a perfect binary tree. Notice that in this case the combined light tree as a whole is not necessarily a perfect tree. Furthermore, two level tree introduces complexity in light tree traversal, but it can improve the sampling quality. An additional benefit of a two-level light tree is that it allows instancing.

4.3.2 Light Tree Traversal

During rendering, we first select a cut through the light tree, similar to the original lightcuts method [26]. Yet, unlike lightcuts, we cannot afford to pick a cut with hundreds of nodes, which would result in too many light samples to achieve real-time performance. Instead, we pick a cut with a relatively small, user-defined number of nodes. Therefore, cut selection is not performed until convergence with a given error threshold. In fact, given a small number of nodes for the cut, it is almost guaranteed that we can never satisfy a reasonable error threshold. Thus, we do not use an error threshold parameter and our cut selection terminates until a user-defined number of nodes (i.e., subtrees) are selected.

Given any subtree root, it is easy to select a light sample by traversing our perfect light tree down to a leaf node. We use the hierarchical importance sampling approach of stochastic lightcuts [4]. At each internal node, one of its child nodes is randomly selected based on their weights computed at the shaded point. To improve the sampling quality in our test scenes, we introduce a novel weight computation scheme for hierarchical importance sampling.

Ideally, the weights should be proportional to the expected illuminations of the child nodes. However, when the shaded point is inside the bounding box of a node, its expected illumination goes to infinity. Yuksel [4] solves this problem by effectively ignoring the distance term when the shaded point is too close to either one of the child node bounding boxes. We use a different strategy. Since the expected illumination can go to infinity, we approximate the expected weights by computing two weights at two different distances from the shaded point: the closest distance $d_j^{\min}$ and the farthest distance $d_j^{\max}$ within the bounding box of the child node j, such that

$$w_j^{\min} = \frac{F_j(\vec{x},\vec{\omega})\left\|\vec{I}_j\right\|}{\left(d_j^{\min}(\vec{x})\right)^2} \qquad w_j^{\max} = \frac{F_j(\vec{x},\vec{\omega})\left\|\vec{I}_j\right\|}{\left(d_j^{\max}(\vec{x})\right)^2}, \tag{4.1}$$

where $\vec{x}$ is the shaded point, $F_j(\vec{x},\vec{\omega})$ is the reflectance bound, and $\vec{I}_j$ is the total light intensity within the node. Note that when the child node bounding boxes are relatively small and far from the shaded point, $w_j^{\min}$ and $w_j^{\max}$ approach to the same values. Given two child nodes j and k, we compute two probabilities for picking j as

$$p_j^{\min} = \frac{w_j^{\min}}{w_j^{\min} + w_k^{\min}} \qquad p_j^{\max} = \frac{w_j^{\max}}{w_j^{\max} + w_k^{\max}}. \tag{4.2}$$

We use the average of these two $p_j = (p_j^{\min} + p_j^{\max})/2$ as the probability of picking child node j. The only singularity with this approach is when both $d_j^{\min}$ and $d_k^{\min}$ are zero (i.e $\vec{x}$ is within the bounding box of both child nodes), in which case we ignore the distance terms for computing $w_j^{\min}$ and $w_k^{\min}$. In our tests, we have found that this new weight computation scheme can improve the quality of light sampling.

Note that a bogus node (or a bogus light) has zero probability to be selected due to its zero intensity. When a *dead branch* is detected, we simply return a light with zero intensity. The intensity of the selected light is divided by the selection probability, and shadows are computed via ray tracing on the GPU.

4.3.3 Light Sampling with Cut Sharing

Cut selection of lightcuts performed independently for each pixel can be expensive. Yet, neighboring pixels often share the same cuts. This is particularly the case when the number of light samples (i.e., the number of nodes above the cut) is small. Based on this observation, we can accelerate cut selection by performing it for a group of pixels, rather than independently for each pixel. Thus, a group of nearby pixels share the same cut through the light tree. We call this *cut sharing*. Note that cut sharing does not cause sampling correlation, since pixels within a group are still free to pick different light samples using the same cut.

Cut sharing can be broken down into two passes. First, a cut computation pass is executed once for each $k \times k$ pixel block where one of the pixels is randomly selected as the representative pixel whose geometric and material properties are used to select the cut.

In the second pass, we perform light sampling, such that each pixel in the $k \times k$ block uses the same cut.

Cut sharing accelerates light sampling with a smaller additional memory footprint for storing the cut. Notice that our cut sharing method bears some resemblance to Adaptive Direct Illumination Sampling [38], where a cut is shared by each scene cell in a 3D grid. In comparison, our screen-space cut sharing enables efficient GPU implementation and screen-space subsampling as explained below.

4.3.4 Interleaved Sampling

Since the cut sharing technique partitions all scene lights into disjoint light clusters (i.e., light subtrees under the cut) for every $k \times k$ pixel block, interleaved sampling of many lights [140, 141] can be naturally used here, such that each pixel in an $m \times m$ subblock only samples from a subset of all light clusters, where $m \leq k$ and k is a multiple of m. Note that, though we use square blocks and subblocks, they can be rectangular as well. The lighting contribution from the lights within a subblock is later shared with the neighboring pixels in a reconstruction process.

Interleaved sampling [140, 141] partitions all light sources into m^2 subsets. Each subset is assigned to one pixel within a block of $m \times m$ pixels. The lighting for each pixel is computed using its subset, reducing the number of light sources used per pixel by $1/m^2$. After computing lighting, a blurring kernel filters the irradiance buffer to remove the structured noise artifacts, using a discontinuity buffer to avoid blurring across geometric edges.

In our method, we instead partition the light subtrees under the chosen cut. Each pixel within a subblock of $m \times m$ pixels receives a subset of the light subtrees and samples only those subtrees. Note that both interleaved sampling (using a subset of the light subtrees) and light sampling (randomly picking lights within each subtree) lead to noise in lighting estimation. Therefore, a simple blurring kernel (like a Gaussian filter used in prior work) may not be sufficient to clear the noise. Instead a geometry-aware denoiser, like SVGF [2], can be applied using more aggressive parameters (such as more blending passes) than typical settings used without interleaved sampling.

The benefit of interleaved sampling is that it can effectively increase the number of light clusters (i.e., the number of nodes along the cut) for a local pixel neighborhood without increasing the number of light samples per pixel. This way, we can process a deeper cut, which often leads to higher quality. Furthermore, deeper cuts would also mean fewer steps needed for hierarchical importance sampling to reach a leaf node, which improves the sampling performance.

4.4 Implementation Details

For generating our perfect light tree (or a bottom-level tree when using a two-level light tree), we first sort all light primitives (points or triangles) based on their centroid positions, using a 30-bit Morton code with 10 bits for each of the x, y, and z coordinates. The centroid positions are quantized using the bounding box of all light sources in the tree (i.e., the global light bounds). We use parallel bitonic sort [144] on 64-bit key-value pairs to produce a sorted index list. This sorted index list is used to populate the leaf level. In our implementation we limit the maximum number of light primitives in each leaf node to 1.

After generating the leaf level, we generate the internal levels in groups. All levels in a group are built in parallel. Given source level ℓ_s, and d destination levels $\ell_s + 1$ to $\ell_s + d$, each destination level is built directly from the source level ℓ_s. Thus, each destination node at level ℓ is formed by merging $2^{\ell - \ell_s}$ source nodes.

In our implementation, the nodes of perfect binary trees do not store light orientation bounding cones. While this overestimates the geometry term used for computing cluster error bounds and sampling weights, we observed only subtle loss of sampling quality in our tests. On the other hand, skipping the cone storage allows us to pack the entire data per light tree node into 32 bytes and helps to align the light tree levels to 64-byte cache lines.

For cut sharing, we use $k = 8$ as the block size for a balance between speed and quality. In our test, we have observed that using a block size smaller than 8 introduced a significant overhead, resulting in slower rendering than not using cut sharing. On the other hand, using a larger block size than 8 rapidly increased the MSE with only minor improvement in sampling time (Figure 4.3). With a block size of 8, the interleaved sampling subblock size m can be chosen as 2, 4, or 8.

Using n light samples per pixel, we pick a cut with n subtrees shared by an entire block.

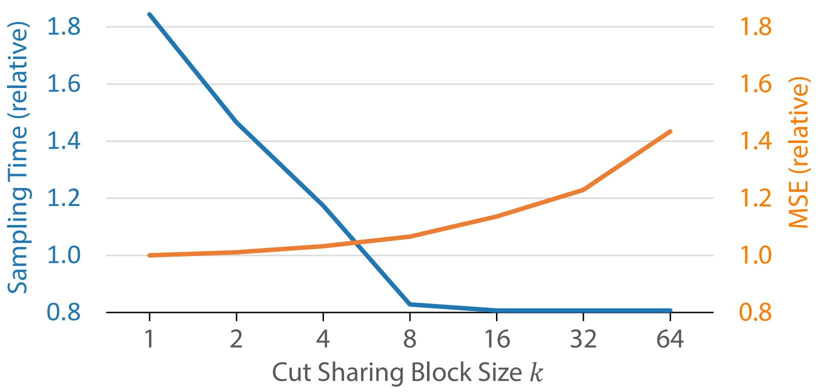


Figure 4.3: The relationship between cut sharing block size k and sampling time (blue line) and MSE (orange line), generated using the Arnold Building scene. The values are relative to the sampling time and MSE of the image rendered without cut sharing (lower is better).

Our interleaved sampling implementation distributes the n subtrees to the pixels within a subblock as evenly as possible. Each pixel with index p within an $m \times m$ subblock (i.e., $p \in \{0, 1, ...m^2 - 1\}$) uses the subtrees t_p through $t_{p+1} - 1$, where $t_p \in \{0, 1, ...n - 1\}$, such that

$$t_p = \left\lfloor \frac{pn}{m^2} \right\rfloor . \tag{4.3}$$

In our implementation, we also rotate the light cluster assignment of pixels within a subblock, so that the same pixel does not get the same clusters every frame. This allows achieving better quality with spatiotemporal filtering. At each frame f, each pixel i within a subblock is assigned an index of $p = i + f \pmod{m^2}$.

4.5 Results

We present test results in different scenes with many lights. Our results do not include interleaved sampling (Section 4.3.4), unless otherwise stated. We compute the flux of textured emitters using the method introduced by Moreau and Clarberg [42]. All scenes but Lumberyard Bistro use a one-level light tree. The *Amazon Lumberyard Bistro* scene uses a two-level light tree mentioned in Section 4.3.1 to improve the sampling quality since the mesh lights have more irregular shapes than other scenes. All scenes only evaluate direct lighting from the light sources or virtual lights. Direct lighting samples are only generated by sampling the lights, without multiple importance sampling. All random numbers used for sampling are generated using the GPU Tiny Encryption Algorithm [145]. The results are rendered at 1920 $\times$ 1080 resolution using an NVIDIA RTX 2080 graphics card on a computer with an Intel Core i7-8700K CPU and 16GB RAM. We implemented our algorithm using the Direct3D 12 graphics API with DirectX Raytracing (DXR) capability. All timings are averaged over 512 frames.

4.5.1 Light Tree Construction

We present the breakdown of light tree construction time in Table 4.1 using two scenes. In the *Crytek Sponza* scene, we generate approximately 100,000 virtual point lights from a single sun (point) light by periodically varying the sun angle via ray tracing in every frame. The *Cornell Box* scene includes 12 animated mesh lights with 12,146 emissive triangles. The bounding box of all lights is first computed using parallel reduction. Notice that sorting is the bottleneck of our light tree construction.

Our light tree construction on the GPU is more than two orders of magnitude faster than the agglomerative clustering on the CPU that produces an unbalanced tree. Agglomerative clustering takes 251 ms for Crytek Sponza and 22 ms for Cornell Box on CPU, while our tree construction takes only 0.43 ms and 0.15 ms on the GPU, respectively.

Note that it is possible to construct an unbalanced light tree using divisive clustering [146], which splits nodes by spatial median in a top-down order. Divisive clustering is faster than agglomerative clustering and more suitable for a GPU implementation. Nonetheless, divisive clustering is not expected to produce a light tree with higher quality that would lead to lower noise in light sampling, as compared to agglomerative clustering. Also, since it would produce an unbalanced tree, it would not have the sampling efficiency of our perfect trees, even it could be optimized to achieve construction speeds closer to ours.

Table 4.1: Breakdown of the light tree construction time.

	Crytek Sponza		*Cornell Box*	
Geometry update	N/A	0%	0.01 ms	6%
Compute bounds	0.03 ms	7%	0.02 ms	10%
Morton code generation	0.01 ms	3%	0.01 ms	4%
Sorting	0.28 ms	63%	0.07 ms	45%
Building the leaf level	0.03 ms	8%	0.01 ms	5%
Building internal levels	0.08 ms	19%	0.05 ms	29%
Total Time	**0.43 ms**	100%	**0.15 ms**	100%

4.5.2 Comparison to Prior Work

We compare our method to a real-time implementation of adaptive tree splitting (ATS) [3]. The light BVH for ATS is built and updated on the CPU using surface area orientation heuristic. Our method uses a single-level perfect light tree, which is rebuilt every frame on the GPU.

One advantage of our method is that by improving the efficiency of light sampling, we can use more light samples. This is demonstrated in the example in Figure 4.4. Using (approximately) the same light sampling time (excluding the light tree construction time) in this scene, our method can process 11 light samples during the time it takes for ATS to process 5 light samples per pixel. As a result, the noise (prior to filtering) is substantially less with our method.

Another example comparing our method to ATS for equal sampling time (excluding light tree construction time) is shown in Figure 4.5. Again, our method is able to process more light samples within the same sampling time, resulting in lower noise. Figure 4.6 shows a close-up of the same scene before and after filtering. Notice that the noise with ATS is not entirely eliminated after filtering and presents itself as lower-frequency noise.

The comparison results are summarized in Table 4.2, along with ATS results using equal sample count. Notice that for the *Crytek Sponza* scene the light tree update time for ATS

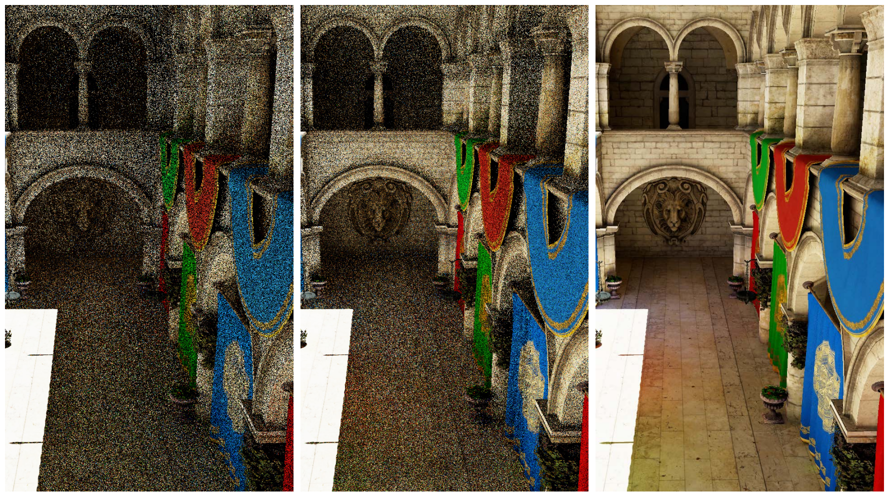
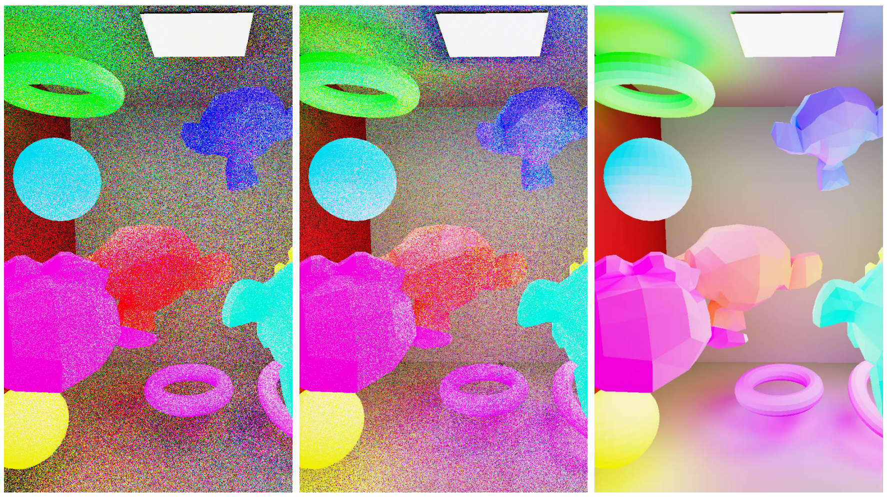

(a) ATS (5 spp)
21.9 ms
(b) Ours (11 spp)
22.1 ms
(c) Reference
5 min

Figure 4.4: Equal sampling time (excluding construction time) comparison between (a) a real-time implementation of Adaptive Tree Splitting (ATS) [3] and (b) our method, using a dynamic lighting condition with 100,000 VPLs in the *Crytek Sponza* scene.

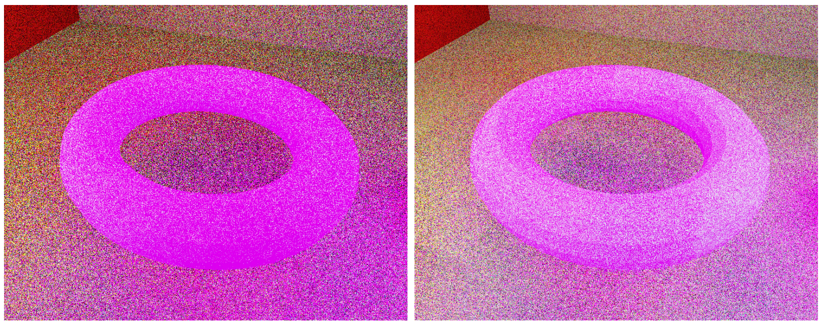


(a) ATS (4 spp)
18.4 ms

(b) Ours (13 spp)
18.1 ms

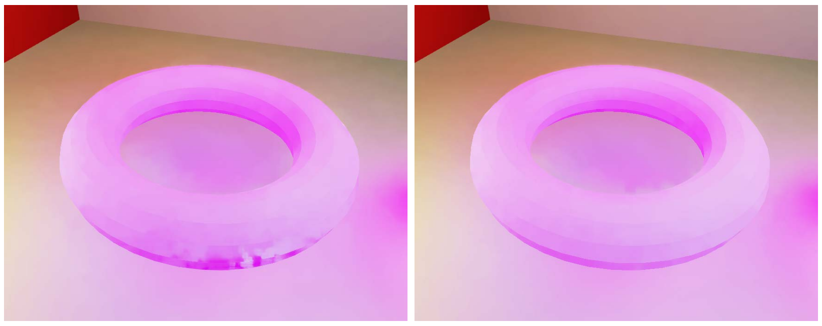
(c) Reference
4 min

Figure 4.5: Equal sampling time (excluding light tree construction time) comparison between (a) a real-time implementation of Adaptive Tree Splitting (ATS) [3] and (b) our method, with dynamic illumination in the *Cornell Box* scene.



(a) ATS (unfiltered)

(b) Ours (unfiltered)



(c) ATS (filtered)

(d) Ours (filtered)

Figure 4.6: Equal sampling time ($\approx$18 ms) comparison between (a) a real-time implementation of Adaptive Tree Splitting (ATS) [3] and (b) our method in the *Cornell Box* scene. (c-d) the bottom row shows the results after filtering with SVGF (α = 0.2) [2]. Noise on the lower part of the torus in the filtered ATS result is apparent even after temporal accumulation of samples.

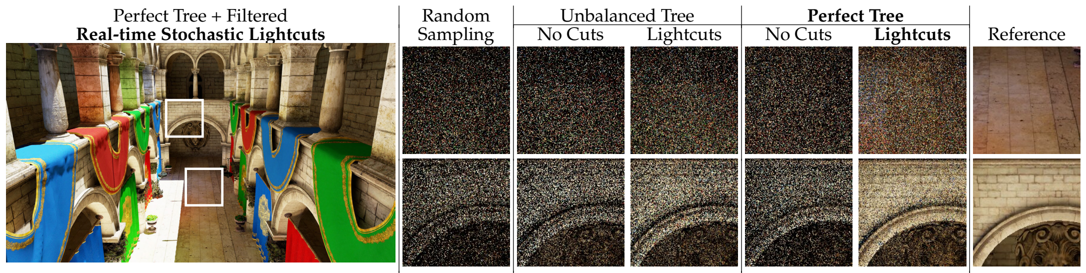
Table 4.2: Comparison to real-time Adaptive Tree Splitting (ATS).

		Light Tree Update	Sampling Time	Sample Count	Average MSE	Average SSIM
	ATS	*CPU:* 176.6 ms	21.9 ms	5	0.090	0.223
	ATS	*CPU:* 176.6 ms	48.1 ms	11	0.058	0.277
	Ours	*GPU:* 0.4 ms	22.1 ms	11	0.064	0.254
	ATS	*CPU:* 0.6 ms	18.4 ms	4	0.084	0.168
	ATS	*CPU:* 0.6 ms	58.9 ms	13	0.033	0.222
	Ours	*GPU:* 0.2 ms	18.1 ms	13	0.040	0.200

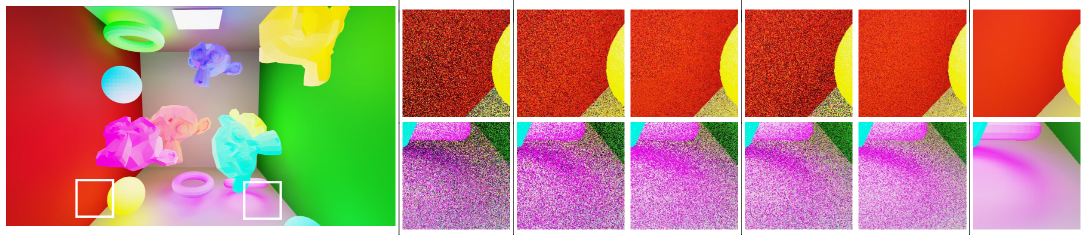
on the CPU is orders of magnitude slower than our light tree construction on the GPU. For the *Cornell Box* scene, on the other hand, we use a two-level light tree with ATS and only the small top-level tree is updated every frame and the bottom-level trees (one for each mesh light) remain constant, significantly reducing the light tree update time of ATS. In comparison, our light tree construction still works multiple times faster, even though it rebuilds an entire (single-level) light tree for every frame from scratch on the GPU. The table also includes mean square error (MSE) and structural similarity index (SSIM) values. Note that the tree quality of ATS allows it to achieve lower noise (i.e., lower MSE and SSIM). This advantage of ATS is due to the fact that we use a perfect tree that delivers lower sampling quality. On the other hand, ATS requires substantially longer sampling time to process as many samples as our method.

4.5.3 Evaluation of Light Sampling Strategies

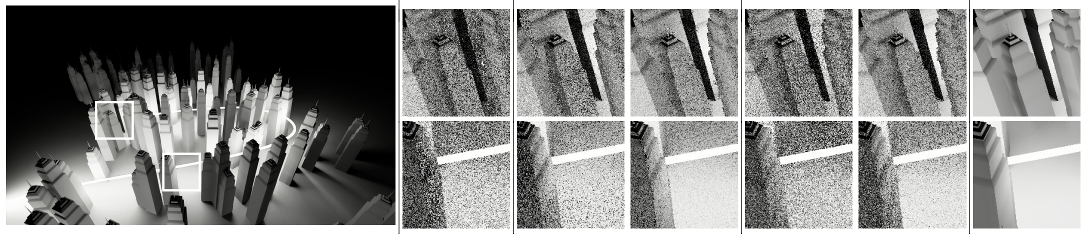
In Figures 4.7 and 4.8 we present (approximately) equal sampling time comparisons of our real-time stochastic lightcuts method to two alternatives: uniform *random sampling* without a light tree and sampling directly from a light tree with *no cuts* (i.e., without selecting a cut). Sampling with no cuts uses hierarchical importance sampling, starting from the root of the tree for each light sample, using our weight computation scheme. Note that directly sampling a light tree is the approach used in some recent work [42, 3]. In comparison, our method uses stochastic *lightcuts* with cut sharing; thus, hierarchical importance sampling traverses the selected subtrees. In these figures we also use two alternatives for light trees: *unbalanced tree* generated using agglomerative clustering on the CPU and *perfect tree* generated using our method on the GPU. The nodes of the unbalanced trees contain orientation bounding cones and child index offsets, which provide



Perfect Tree + Filtered **Real-time Stochastic Lightcuts**		Random Sampling	Unbalanced Tree No Cuts	Unbalanced Tree Lightcuts	**Perfect Tree** No Cuts	**Perfect Tree** **Lightcuts**	Reference
Crytek Sponza	**Build Time:**	- - -	*CPU:* 251 ms	*CPU:* 251 ms	*GPU:* 0.4 ms	*GPU:* 0.4 ms	
261,798 triangles	**Sampling Time:**	31.3 ms	29.4 ms	29.1 ms	29.3 ms	29.7 ms	
100,000 virtual point lights	**Light Samples:**	12	6	9	7	17	
	MSE:	0.108	0.098	0.078	0.097	0.050	
	SSIM:	0.181	0.194	0.222	0.195	0.285	

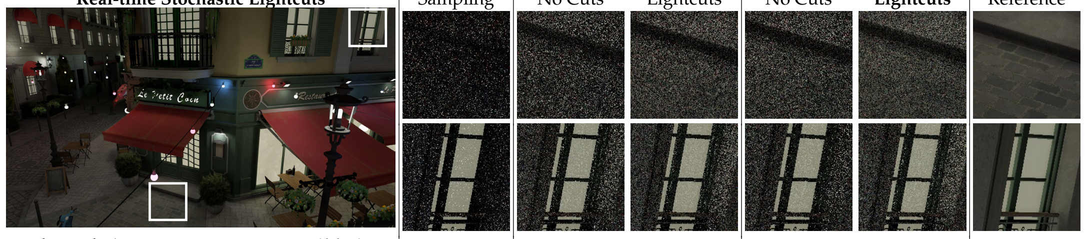


Perfect Tree + Filtered **Real-time Stochastic Lightcuts**		Random Sampling	Unbalanced Tree No Cuts	Unbalanced Tree Lightcuts	**Perfect Tree** No Cuts	**Perfect Tree** **Lightcuts**	Reference
Cornell Box	**Build Time:**	- - -	*CPU:* 22.4 ms	*CPU:* 22.4 ms	*GPU:* 0.2 ms	*GPU:* 0.2 ms	
12,156 triangles	**Sampling Time:**	29.9 ms	29.4 ms	30.8 ms	30.4 ms	30.5 ms	
12,146 triangle lights	**Light Samples:**	30	9	14	16	23	
	MSE:	0.055	0.048	0.028	0.036	0.023	
	SSIM:	0.181	0.178	0.217	0.193	0.238	



Perfect Tree + Filtered **Real-time Stochastic Lightcuts**		Random Sampling	Unbalanced Tree No Cuts	Unbalanced Tree Lightcuts	**Perfect Tree** No Cuts	**Perfect Tree** **Lightcuts**	Reference
Arnold Buildings	**Build Time:**	- - -	*CPU:* 8.1 ms	*CPU:* 8.1 ms	*GPU:* 0.1 ms	*GPU:* 0.1 ms	
94,206 triangles	**Sampling Time:**	30.4 ms	30.4 ms	29.6 ms	30.7 ms	30.2 ms	
4,596 triangle lights	**Light Samples:**	35	12	17	19	29	
	MSE:	0.046	0.021	0.011	0.033	0.019	
	SSIM:	0.439	0.484	0.534	0.457	0.503	

Figure 4.7: Visual and quantitative comparison of light sampling methods with (approximately) equal sampling time. The unbalanced trees are built on the CPU using agglomerative clustering and perfect trees are build using our method on the GPU.



Two-Level Perfect Trees + Filtered **Real-time Stochastic Lightcuts**		Random Sampling	Unbalanced Tree No Cuts	Unbalanced Tree Lightcuts	**Two-Level Perfect Trees** No Cuts	**Two-Level Perfect Trees** **Lightcuts**	Reference
Lumberyard Bistro	**Build Time:**	- - -	*CPU:* 44.4 ms	*CPU:* 44.4 ms	*GPU:* 3.2 ms	*GPU:* 3.2 ms	
2,837,137 triangles	**Sampling Time:**	29.5 ms	30.7 ms	30.0 ms	29.2 ms	30.7 ms	
200 mesh lights	**Light Samples:**	13	6	9	8	16	
24,782 emissive triangles	**MSE:**	0.033	0.025	0.023	0.028	0.022	
	SSIM:	0.160	0.210	0.222	0.194	0.235	

Figure 4.8: Comparison of methods using a similar setup as Figure 4.7.

better sampling quality but some reduction in sampling speed (in addition to the extended build time). The examples in Figure 4.7 use single-level perfect trees and the example in Figure 4.8 uses a two-level perfect tree (the bottom-level perfect trees are built on the CPU in our implementation). The reference images are generated using stochastic lightcuts results with 65,536 total samples per pixel.

Notice that in all four scenes the highest light sample count is achieved by either our stochastic lightcuts method with perfect tree or random sampling. While uniform random sampling cannot effectively make use of the relatively high sample count, our method leads to a relatively low noise solution. The only exception is the *Arnold Buildings* scene, where the quality improvement of using an unbalanced tree makes a significant-enough improvement, so that our stochastic lightcuts method with an unbalanced tree provides the best quality, even by using fewer light samples than our method with a perfect light tree. This shows the performance gain of using a perfect light tree may not always offset the potential reduction in the tree quality. However, this comparison does not include the substantially longer build time of the unbalanced tree. Notice that our method benefits from the fast sampling speed which allows it to use 1.4× - 2.8× more samples than other methods. On the other hand, methods based on unbalanced light trees have prohibitively expensive built time (8.1 ms - 251.0 ms) which makes them impractical to use for fully dynamic scenes.

We provide numerical comparisons using equal sample count (8 light samples per pixel) in Table 4.3, showing total frame render time, sampling time (including cut selection), mean square error (MSE), and structural similarity index (SSIM). Notice that our method with a perfect tree is faster in both total frame time and sampling time than all alternatives, except for random sampling. In fact, our method delivers a higher sampling speed than random sampling in the *Crytek Sponza* scene, which is likely due to more coherent shadow rays. However, the quality improvement over random sampling (as can be seen by the MSE and SSIM numbers) is substantial, as expected.

Also notice that, using 8 light samples per pixel, there is little numerical difference in quality between using lightcuts and directly sampling the light tree with no cuts. Indeed, in the *Arnold Buildings* scene, using no cuts actually provides slightly better quality. This is mainly because, when using relatively few light samples, the selected cut may not always

Table 4.3: Comparisons using 8 light samples per pixel.

	Random Sampling	Unbalanced Tree		**Perfect Tree**	
		No Cuts	Lightcuts	No Cuts	**Lightcuts**
Frame Time					
Crytek Sponza	29.9 ms	300.1 ms	287.9 ms	43.7 ms	27.4 ms
Cornell Box	13.8 ms	53.1 ms	44.7 ms	22.3 ms	19.1 ms
Arnold Buildings	12.7 ms	26.5 ms	21.5 ms	18.9 ms	15.9 ms
Lumberyard Bistro	28.1 ms	52.0 ms	36.9 ms	39.9 ms	28.7 ms
Sampling Time					
Crytek Sponza	20.8 ms	39.3 ms	26.1 ms	33.4 ms	17.4 ms
Cornell Box	7.7 ms	25.7 ms	18.6 ms	14.8 ms	11.8 ms
Arnold Buildings	6.4 ms	19.9 ms	15.1 ms	12.4 ms	9.8 ms
Lumberyard Bistro	18.1 ms	41.0 ms	26.9 ms	29.2 ms	18.7 ms
MSE					
Crytek Sponza	0.119	0.088	0.093	0.092	0.085
Cornell Box	0.123	0.053	0.048	0.070	0.065
Arnold Buildings	0.107	0.030	0.039	0.063	0.065
Lumberyard Bistro	0.035	0.022	0.025	0.028	0.028
SSIM					
Crytek Sponza	0.170	0.208	0.205	0.201	0.214
Cornell Box	0.155	0.173	0.179	0.162	0.165
Arnold Buildings	0.369	0.451	0.436	0.403	0.401
Lumberyard Bistro	0.149	0.232	0.208	0.195	0.188

provide a good clustering of lights and some light trees may have substantially stronger illumination than others. Therefore, using one light sample per subtree, as we use in our stochastic lightcuts approach, does not always provide the best sample distribution. Nonetheless, the performance improvement due to cut sharing allows using more light samples per pixel, thereby improving the final result within the same render time.

Moreover, using an unbalanced tree often improves the sampling quality with the same number of samples. On the other hand, the cost of building an unbalanced tree and sampling it leads to using fewer light samples within the same render time.

4.5.4 Light Sampling Improvements

Our cut sharing method (Section 4.3.3) can provide substantial reduction in sampling time. An example of this is shown in Figure 4.9, comparing sampling quality and performance with and without cut sharing. Notice that images with cut sharing can include visual artifacts prior to filtering, particularly near depth discontinuities (Figure 4.9a). Yet,

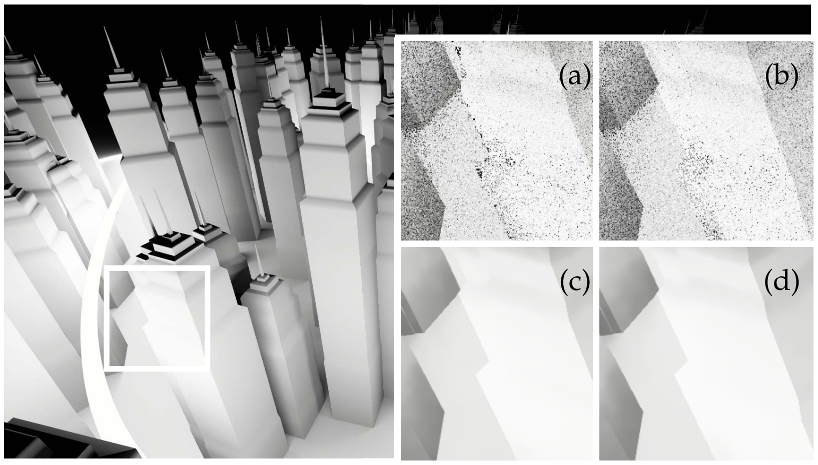
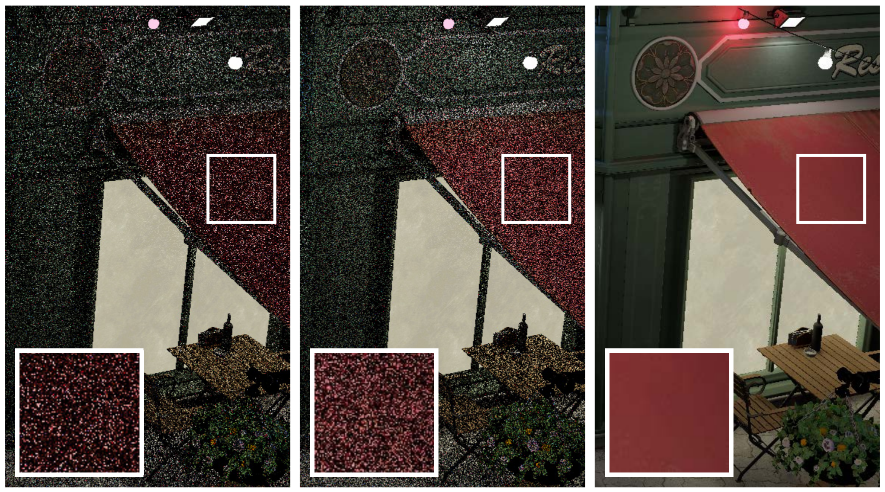


Figure 4.9: Images rendered with and without cut sharing using 32 light samples per pixel: (a) cut sharing with $k = 8$ (sampling time: 31.3 ms) before filtering, (b) no cut sharing (sampling time: 37.8 ms) before filtering, (c) cut sharing after filtering, and (d) no cut sharing after filtering. The filtered results are generated using an SVGF filter [2] with $\alpha = 0.2$ and accumulation of 5 frames.

after applying spatiotemporal filtering, it produces indistinguishable filtered results (Figure 4.9c). In terms of performance, cut sharing provides about 20% speedup in sampling time in this example, including the overhead of the cut selection computation, which takes less than 2% of the light sampling time.

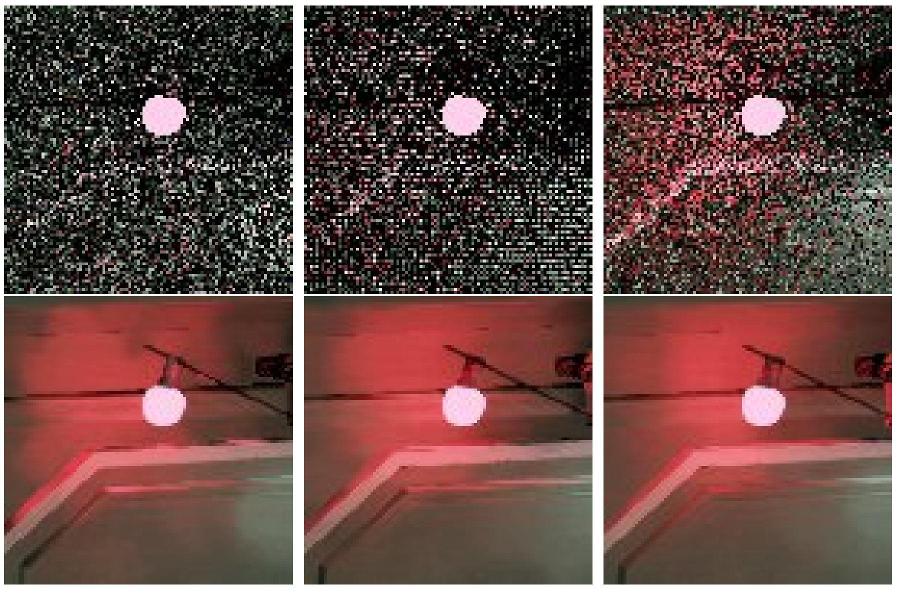
As we explain in Section 4.3.1 our method permits using two-level light trees. Using a two-level light tree introduces some additional cost, both in light tree construction and light sampling. On the other hand, depending on the scene, the improvement over using a single perfect tree can be substantial. We demonstrate this in Figure 4.10 using the *Amazon Lumberyard Bistro* scene. In this case, using a two-level light tree slightly increases the light sampling time from 17.4 ms to 18.7 ms, but it also reduces the noise. A two-level tree is expected to provide some improvement in quality and some reduction in performance, but whether a two-level tree would be beneficial over a single-level perfect tree depends on the scene.

As we explain in Section 4.3.4, our cut sharing approach allows using interleaved sampling for further reducing the sampling cost or increasing the effective sample count. The example in Figure 4.11 shows the *Amazon Lumberyard Bistro* scene rendered with and without interleaved sampling, using 8 samples per pixel. In this case, interleaved sampling uses a subblock size of 2×2. Therefore, each cut for each subblock contains 32



(a) Single-Level (b) Two-Level (c) Reference

Figure 4.10: The quality improvement of using a two-level tree: images for both (a) single-level and (b) two-level trees are rendered using stochastic lightcuts with 8 light samples per pixel.



(a) No Inter. Samp. 8 spp 18.7 ms (b) Interleaved Samp. 8 spp (2×2) 21.0 ms (c) No Inter. Samp. 32 spp 56.1 ms

Figure 4.11: Interleaved sampling: (a) no interleaved sampling with 8 samples per pixel, (b) interleaved sampling within a 2×2 subblock with 8 samples per pixel, approximating 32 samples per pixel, and (c) no interleaved sampling with 32 samples per pixel. The images on the bottom row show results filtered results with SVGF [2] without temporal accumulation (to amplify the difference of quality). The images are rendered using the screen and camera configuration of *Lumberyard Bistro* in Figure 4.8.

light samples. As a result, interleaved sampling can approximate the quality of using 32 light samples by computing 8 light samples per pixel. Note that interleaved sampling is particularly effective when combined with spatiotemporal filtering.

Figure 4.12 compares the performance of hierarchical importance sampling using the new weight computation method we introduced in Section 4.3.2 to the weight computation scheme in the stochastic lightcuts paper [4]. Notice that in all our test scenes our new weights computation provides a minor but visible improvement in sampling quality.

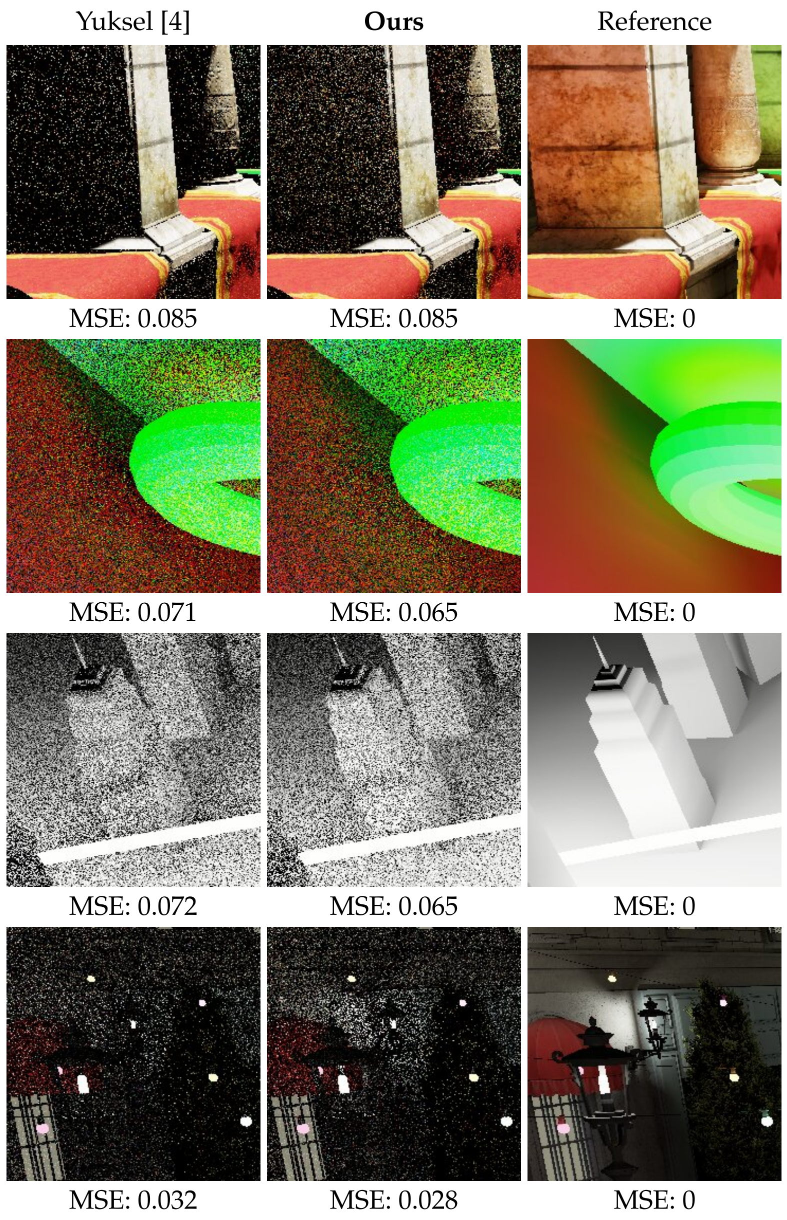


Figure 4.12: Hierarchical importance sampling using the weight computation scheme of Yuksel [4] and our new weight computation method (Section 4.3.2).

4.6 Limitations

While our perfect binary trees can be built extremely fast on the GPU using Morton codes, the resulting light trees lead to more noise as compared to unbalanced light trees generated with agglomerative clustering. This is because using a balanced tree and only considering spatial proximity when partitioning the nodes limits the quality of the light tree. Although the improvement in sampling speed with perfect trees can compensate for the loss in tree quality in our tests, they might be less effective in some scenes.

While our cut sharing technique has been effective in our tests, it might lead to visual artifacts in some scenes. In particular, specular highlights on highly glossy surfaces near cut sharing block boundaries may form discontinuities that would be difficult to detect with a geometry-aware filter. Furthermore, since our cut sharing method uses screen-space blocks, the chosen cut for a block may not be ideal for all pixels of the block, particularly near depth discontinuities, and it is not suitable for more general applications, like path tracing or ray traced reflections.

4.7 Conclusion

We have presented a real-time light sampling technique for scenes with many lights. Our method extends stochastic lightcuts by using a perfect binary light tree with a novel weight computation scheme and cut sharing. By minimizing the cost of light sampling, our method allows using more light samples within the same render time for achieving higher sampling quality. Since our method does not restrict the types of lights that can be sampled and the light tree can be constructed efficiently every frame, we can accommodate fully dynamic scenes with a variety of light types.

CHAPTER 5

FAST VOLUME RENDERING WITH SPATIOTEMPORAL RESERVOIR RESAMPLING[1]

Volume rendering under complex, dynamic lighting is challenging, especially if targeting real-time. To address this challenge, in this chapter, we extend a recent direct illumination sampling technique, spatiotemporal reservoir resampling, to multi-dimensional path space for volumetric media.

By fully evaluating just a single path sample per pixel, our volumetric path tracer shows unprecedented convergence. To achieve this, we properly estimate the chosen sample's probability via approximate perfect importance sampling with spatiotemporal resampling. A key observation is recognizing that applying cheaper, biased techniques to approximate scattering along candidate paths (during resampling) does not add bias when shading. This allows us to combine transmittance evaluation techniques: cheap approximations where evaluations must occur many times for reuse, and unbiased methods for final, per-pixel evaluation.

With this reformulation, we achieve low-noise, interactive volumetric path tracing with arbitrary dynamic lighting, including volumetric emission, and maintain interactive performance even on high-resolution volumes. When paired with denoising, our low-noise sampling helps preserve smaller-scale volumetric details.

5.1 Introduction

Smoke, fire, clouds, and other participating media are vital in virtual scenes; modern movies, games, and simulations rely heavily on media for realism and ambiance. But

[1]D. Lin, C. Wyman, and C. Yuksel, "Fast volume rendering with spatiotemporal reservoir resampling," ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2021), vol. 40, no. 6, pp. 279:1–279:18, 2021.

real-time rendering of participating media is challenging. Even traditional raster pipelines have separate order-independent transparency passes [147] and data structures for volume lighting [50]. With real-time ray tracing [148] and more complex ray-traced lighting [72], integrating dynamic ray-traced media will be vital for achieving a uniform look.

In this chapter, we introduce an effective path sampling solution for real-time volume rendering with multiple scattering and volumetric emission. To do this, we generalize *resampled importance sampling* [17] and *spatiotemporal reservoir resampling* [7] to path integrals. These have proven effective for sampling direct illumination on surfaces. We generalize them to path space, providing importance sampling that closely approximates our integrand: a path integral formulation of the volume rendering equation. We also present numerous optimizations to minimize computation and memory overhead. As a result, we can estimate the multi-dimensional volume rendering integral while shading just one path per pixel, enabling real-time volume rendering under arbitrary scene illumination, including environment maps, area lights, and volumetric emission.

Our technical contributions include:

- A generalization of resampled importance sampling (Section 5.3) and spatiotemporal reservoir resampling (Section 5.4) to complex path integrals,
- An efficient importance sampling estimator for the volumetric path integral (Section 5.3.3), including multiple scattering and volumetric light emission,
- A temporal reprojection (Section 5.4.4) and practical velocity resampling method for robust temporal reuse (Section 5.5.2),
- Optimized path space transmittance estimates (Section 5.5.1). These only affect importance sampling, not final path throughput, allowing use of efficient biased estimates without biasing the results (e.g., sampling lower resolution volumes).

Our renderer runs interactively, reusing carefully-chosen paths to evaluate the volume rendering equation. While we can produce unbiased renderings (with static volumes and dynamic lighting), by allowing a little bias (Section 5.4.4) we can reduce sampling variance and handle more dynamism. Our work significantly lowers variance compared to state-of-the-art real-time path sampling (Figure 5.1).

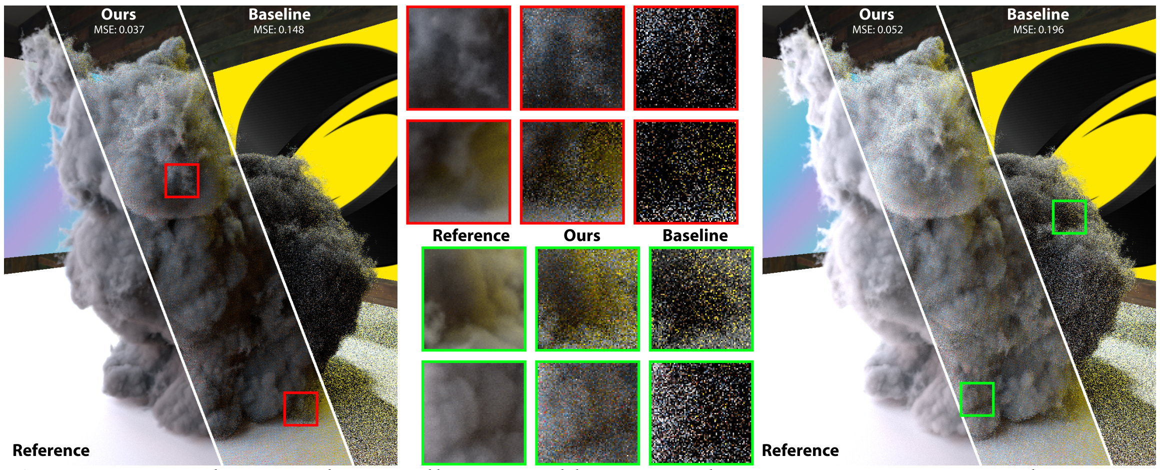


Figure 5.1: A volumetric bunny illuminated by a complex environment map and emissive logos. We compare our new volumetric ReSTIR with offline references and an equal-time baseline (combining decomposition tracking [5] and residual ratio tracking [6]). We show our work with (left) single scattering in 55 ms and (right) three-bounce multiple scattering in 142 ms.

Section 5.2 reviews prior work, summarizing resampled importance sampling (RIS) and spatiotemporal reservoir resampling (ReSTIR). In Section 5.3, we develop an RIS estimator to efficiently sample the volumetric path integral. In Section 5.4, we modify ReSTIR's iterative resampling to efficiently reuse spatiotemporal volumetric path samples. We provide key implementation details in Section 6.7.

5.2 Background: RIS and ReSTIR

In this section, we provide a detailed introduction of resampled importance sampling (RIS) and ReSTIR, which helps the readers to understand the basis of our method. Note that the notations here follow Bitterli et al. [7]. We re-introduce RIS and ReSTIR using "proper" notations that fit into our generalized resampled importance sampling framework in Chapter 6.

5.2.1 Resampled Importance Sampling (RIS)

Our volume sampling builds on *resampled importance sampling* (RIS) [17], Given function $f(x)$ defined over domain $x \in D$, RIS provides an importance sampling estimator for the integral:

$$I = \int_D f(x)\, dx\,. \tag{5.1}$$

Let $\hat{p}$ be a *target* PDF without a practical sampling algorithm. RIS generates $M \geq 1$ *candidate samples* $= \{x_1, \dots, x_M\}$ using a (suboptimal) *source* PDF p. Then, it randomly selects a sample x_r, for $r \in \{1, \dots, M\}$, using discrete probabilities

$$p(x_r|) = \frac{w(x_r)}{\sum_{j=1}^{M} w(x_j)} \qquad \text{with} \qquad w(x) = \frac{\hat{p}(x)}{p(x)} \,. \tag{5.2}$$

The resulting 1-sample RIS estimator can be written as

$$\langle I \rangle_{\text{ris}}^{1,M} = E_{\hat{p}}(x_r) \left(\frac{1}{M} \sum_{j=1}^{M} w(x_j) \right) \qquad \text{with} \qquad E_{\hat{p}}(x_r) = \frac{f(x_r)}{\hat{p}(x_r)} \,. \tag{5.3}$$

The parenthetical term corrects for differences between the actual probability used to sample x_r and the desired PDF $\hat{p}(x_r)$. This gives an unbiased estimate if $p(x)$ and $\hat{p}(x)$ are non-zero for all x with non-zero $f(x)$. As $M \to \infty$, the distribution of x_r approaches $\hat{p}$.

RIS is particularly effective if $\hat{p}$ closely approximates f and generation and evaluation of candidate samples x_j and $w(x_j)$ are cheap. In Talbot et al. [17], x is a point on a light source, $p(x)$ is the light sampling PDF. $\hat{p}(x)$ is unshadowed reflected radiance, including BSDF, geometry term, and incident radiance. This reasonably approximates the integrand, without expensive visibility queries. When directly lighting opaque surfaces, this improves sampling quality over standard importance sampling.

5.2.2 Spatiotemporal Reservoir Resampling (ReSTIR)

Spatiotemporal reservoir resampling (ReSTIR) [7] transforms RIS into a streaming algorithm, avoiding storage of most candidate samples by using weighted reservoir sampling [121]. It is designed for direct illumination sampling from many lights for real-time rendering. For each pixel, ReSTIR maintains a reservoir that stores a sample x_r selected from the previous m candidates. Each new candidate x_{m+1} is selected with probability

$$p(x_{m+1}| \cup \{x_{m+1}\}) = \frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)} \,. \tag{5.4}$$

This can be seen as *streaming* candidate samples into a reservoir. Since the reservoir stores only the selected sample and a running sum of weights $\sum_{j=1}^{m+1} w(x_j)$, many candidate samples M can be considered without additional storage, improving the sampling quality.

ReSTIR also enables spatiotemporal reuse by combining the reservoirs of nearby pixels and the previous frame, exponentially increasing the *effective* candidate sample count (Figure 5.2). While rendering the first frame, each pixel q allocates a reservoir and streams M

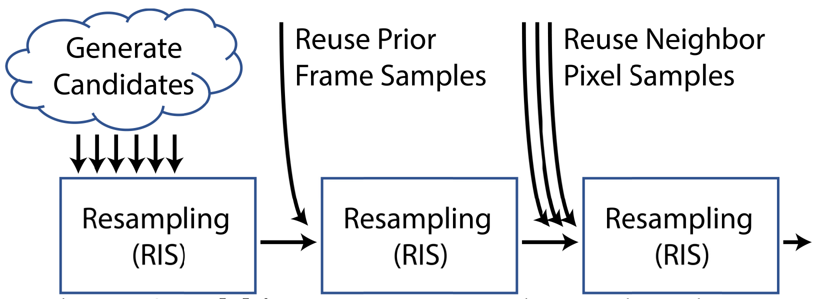


Figure 5.2: Extending ReSTIR [7] for participating media entails updating each component of this ReSTIR pipeline: defining candidate generation in path space (see Section 5.3.2), RIS estimators to efficiently evaluate high-dimensional integrals (Section 5.3.3), and spatial and temporal sample reuse in this domain (Section 5.4).

newly generated candidates to select a sample x_q. Then, the reservoirs of a random subset of nearby pixels are combined. Let q' represent a pixel near q. Their two reservoirs cannot simply be combined, unless the target PDFs $\hat{p}_q(x)$ and $\hat{p}_{q'}(x)$ for both pixels are identical. Generally, this is not the case and $\hat{p}_q(x) \neq \hat{p}_{q'}(x)$. Therefore, ReSTIR includes a correction factor $w_{q' \to q}$ for using the selected sample $x_{q'}$ in reservoir q' for pixel q, defined as

$$w_{q' \to q} = \frac{\hat{p}_q(x_{q'})}{\hat{p}_{q'}(x_{q'})} w_{q'}^{\text{sum}} , \qquad \text{where} \qquad w_{q'}^{\text{sum}} = \sum_{j=1}^{M} \frac{\hat{p}_{q'}(x_j)}{p_{q'}(x_j)} . \tag{5.5}$$

Note that $w_{q'}^{\text{sum}}$ is the running sum in the reservoir from initial candidate generation. For multiple iterations of reuse (chained RIS passes), $w_{q'}^{\text{sum}}$ becomes the running sum from the prior RIS pass.

The resulting estimator combining N neighboring reservoirs from pixels $q_1, \ldots, q_N$ can be written as

$$\langle I \rangle_{\text{ReSTIR}}^{N,1,M} = E_{\hat{p}_q}(x_r) \left(\frac{1}{M_q} \sum_{i=0}^{N} w_{q_i \to q} \right) , \tag{5.6}$$

where x_r is the sample selected from one of the $N{+}1$ reservoirs, $M_q = (N{+}1)M$ is the total *effective* candidate sample count for pixel q, and we define $q_0 \equiv q$. Yet, this leads to a biased estimator, because $\hat{p}_{q'}(x)$ can be zero for x with non-zero $p_q(x)$. For correcting this bias, Bitterli et al. [7] propose replacing the $1/M_q$ term in Equation 5.6 with the MIS weight of the selected sample

$$\frac{\hat{p}_r(x_r)}{M \sum_{i=0}^{N} \hat{p}_{q_i}(x_r)} , \tag{5.7}$$

where $\hat{p}_r$ denotes the PDF of the reservoir that produced x_r.

This MIS weight is stochastic (it depends on the chosen sample). Although cheaper to evaluate than the deterministic MIS weight proposed by Talbot [8], it introduces noise. The deterministic Talbot MIS can be used by multiplying the weights of samples in Equation 5.6 with an additional term:

$$w^{\text{new}}_{q_i \to q} = w_{q_i \to q} \cdot \frac{M_q \hat{p}_{q_i}(x_{q_i})}{M \sum_{s=0}^{N} \hat{p}_{q_s}(x_{q_i})}\ , \tag{5.8}$$

Additionally, ReSTIR allows temporal reuse by passing the final selected sample x_r forward for reuse next frame. Combining spatial and temporal reuse, the *effective* candidate sample count M_q grows exponentially. To prevent unbounded influence of temporal samples, a user-defined *temporal limiting factor* Q is used to enforce $M_q \leq QM$. When M_q exceeds this limit, the running sum is scaled by QM/M_q and then M_q is updated as $M_q \leftarrow QM$.

But the benefit of spatiotemporal reuse is not indefinite. If $\hat{p}_{q'}$ from neighbor reservoir q' substantially differs from $\hat{p}_q$, spatially reusing q' can negatively impact sampling quality instead of improving it. Thus, applying heuristics to selectively reject reservoirs and using high quality MIS to reweight samples can substantially reduce variance [149, 150].

ReSTIR is highly effective in estimating direct illumination on opaque surfaces from many lights [7]. It chains RIS passes spatiotemporally to quickly accumulate many samples. Additionally, successive RIS passes can use higher quality $\hat{p}$ to improve sampling quality at a relatively low cost. While $\hat{p}$ typically contains unshadowed reflected radiance, ReSTIR injects visibility into $\hat{p}$ at a lower frequency (known as *visibility reuse*).

Boksansky et al. [111] store reservoirs in world space so a path tracer can efficiently perform NEE on secondary path vertices. Concurrent work by Ouyang et al. [10] extends screen space ReSTIR for surface global illumination. This can be viewed as a special case of our method, which handles both surface and volume transport and interreflections between them (see Figure 5.1, right).

We extend the concepts in ReSTIR to real-time volume rendering; to achieve that, we solve various challenges when resampling (Section 5.3) and reusing samples (Section 5.4) in volumetric path space.

5.3 RIS for Volume Rendering

We target extending ReSTIR [7] to volumetric path tracing. As ReSTIR builds on RIS [17], we begin by developing an RIS estimator for the volume rendering equation.

Volume rendering involves higher-dimensional integrals than the direct surface illumination in Bitterli et al. [7] and Talbot et al. [17]. Visibility alone forms an integral along primary rays. Thus, we cannot just sample light positions; we must sample entire paths.

In this section, we provide a path integral representation of the volume rendering equation (Section 5.3.1), describe how we can generate candidate paths (Section 5.3.2), and explain how to estimate the volume rendering equation using RIS (Section 5.3.3).

5.3.1 Path Integral Representation of Volume Rendering

Let $\boldsymbol{\lambda}$ denote a path. We can write the volume rendering equation (Equation 2.1) as a path integral

$$L(\mathbf{x}_0, \boldsymbol{\omega}_o) = \int_{\boldsymbol{\Lambda}} F(\boldsymbol{\lambda})\, d\boldsymbol{\lambda}\ , \tag{5.9}$$

where $\boldsymbol{\Lambda}$ is the set of all paths and $F(\boldsymbol{\lambda})$ is the incident radiance through the path $\boldsymbol{\lambda}$.

Consider a path $\boldsymbol{\lambda}$ with k scattering events, forming $k+2$ vertices $\mathbf{x}_0, \ldots, \mathbf{x}_{k+1}$, with $\mathbf{x}_0$ a camera vertex and $\mathbf{x}_{k+1}$ a *light vertex*. A light vertex can be a point on a light surface or inside emissive media. For brevity, the formulations below assume intermediate vertices $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are in the medium, but this can easily be extended to points on surfaces (e.g., see Figures 5.1).

Let $z_i = |\mathbf{x}_{i+1} - \mathbf{x}_i|$ be the distance between consecutive path vertices and $\boldsymbol{\omega}_i = (\mathbf{x}_{i+1} - \mathbf{x}_i)/z_i$ be the direction towards the next path vertex. Then, we can write the incident radiance as

$$F(\boldsymbol{\lambda}) = \Gamma_s(\boldsymbol{\lambda}, k)\, T(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\, G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\, L(\mathbf{x}_{k+1} \to \mathbf{x}_k), \tag{5.10}$$

where geometry term $G_i = G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)$ is 1 using solid angle measure and $1/z_i^2$ using volume measure, and Γ_s represents path throughput:

$$\Gamma_s(\boldsymbol{\lambda}, k) = \prod_{i=1}^{k} T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)\, \sigma_s(\mathbf{x}_i)\, G_i\, \rho_i\ , \tag{5.11}$$

with $\rho_i = \rho(\mathbf{x}_i, -\boldsymbol{\omega}_{i-1}, \boldsymbol{\omega}_i)$ the phase function and $L(\mathbf{x}_{k+1} \to \mathbf{x}_k)$ the emitted radiance at $\mathbf{x}_{k+1}$ towards $\mathbf{x}_k$. Note, for a path with $k = 0$ (i.e., no scattering events) we take $\Gamma_s(\boldsymbol{\lambda}, k) = 1$.

The emitted radiance L can come from either a light sample at $\mathbf{x}_{k+1}$, if $\boldsymbol{\lambda}$ is a *scattering path*, or volumetric emission at $\mathbf{x}_{k+1}$, if $\boldsymbol{\lambda}$ is an *emission path*. More specifically, we can write

$$L(\mathbf{x}_{k+1} \to \mathbf{x}_k) = \begin{cases} L^s(\mathbf{x}_{k+1} \to \mathbf{x}_k) & \text{if scattering path,} \\ \sigma_a(\mathbf{x}_{k+1})\, L_e^m(\mathbf{x}_{k+1} \to \mathbf{x}_k) & \text{if emission path.} \end{cases} \tag{5.12}$$

We use L^s to represent radiance from the light. The notation assumes the source is an emissive surface, but it can easily be extended to other lights. For example, for an environment map $\mathbf{x}_{k+1}$ is an infinitely distant vertex and L^s is the radiance along direction $\mathbf{x}_{k+1} \to \mathbf{x}_k$.

5.3.2 Generating Path Samples

To use an RIS estimator for the path integral formulation of volume rendering (Equation 5.9), we must generate numerous random paths $\boldsymbol{\lambda}$ with PDF $p(\boldsymbol{\lambda})$.

Our path generation approach is similar to path tracing with next event estimation, as shown in Figure 5.3. We start with a ray from $\mathbf{x}_0$ towards $\boldsymbol{\omega}_0 = -\boldsymbol{\omega}_o$. At each step, we first pick a random distance z_i along our ray with PDF $p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1})$. This specifies the next path vertex $\mathbf{x}_i = \mathbf{x}_{i-1} + z_i\boldsymbol{\omega}_{i-1}$. Then, for each scattering event, we pick a scattering direction $\boldsymbol{\omega}_i$ with a PDF $p(\boldsymbol{\omega}_i|\mathbf{x}_i)$. We repeat this step to generate a random walk of a desired length.

As shown in Figure 5.3, each vertex $\mathbf{x}_i$ on our random walk spawns two candidate paths to feed our resampling. The first is a scattering path, using next event estimation to sample a light for $\mathbf{x}_{i+1}$. If our media emits light, we generate an emission path ending at $\mathbf{x}_i$. Both scattering and emission paths end at a light: either on a surface or in the media. Emission at intermediate vertices is ignored, as it is accounted for on shorter paths (i.e., spawned at vertex $\mathbf{x}_j$, for $j < i$).

With this procedure, the PDF of a scattering or emission path with k scattering events

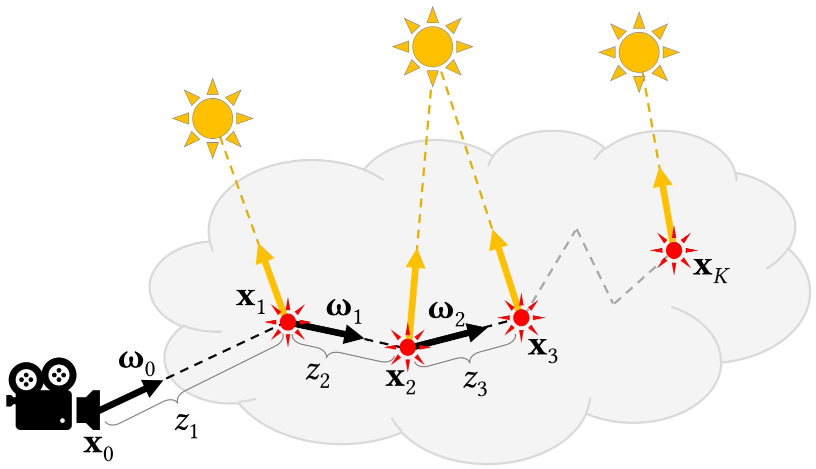


Figure 5.3: A random walk with K vertices generates $2K$ candidate paths for later reuse: K are scattering paths that terminate at a light (yellow) with next event estimation and the remaining K are emission paths terminating in the media due to volumetric emission (red).

and $k+2$ vertices can be written as

$$p(\boldsymbol{\lambda}) = \prod_{i=1}^{k'} p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1})\, G_i\, p(\boldsymbol{\omega}_i|\mathbf{x}_i)\ , \tag{5.13}$$

where $k' = k$ for scattering paths and $k+1$ for emission paths. Here, the PDF of sampling a direction is

$$p(\boldsymbol{\omega}_i|\mathbf{x}_i) = \begin{cases} \rho_i & \text{if } i < k', \\ p_{\text{NEE}}(\boldsymbol{\omega}_k|\mathbf{x}_k) & \text{if } i = k' \text{ and scattering path,} \\ 1 & \text{if } i = k' \text{ and emission path.} \end{cases} \tag{5.14}$$

where $p_{\text{NEE}}(\boldsymbol{\omega}_k|\mathbf{x}_k)$ represents the light sampling PDF used for next event estimation.

The PDF of sampling a distance z_i along a ray from $\mathbf{x}_{i-1}$ towards $\boldsymbol{\omega}_{i-1}$ depends on the sampling method used. When using RIS without ReSTIR, delta tracking [83] is a convenient choice for this task. Delta tracking has a very desirable PDF

$$p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1}) = T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)\, \sigma_t(\mathbf{x}_i)\ . \tag{5.15}$$

A problem with this PDF is transmission T is either not available in closed form or expensive to compute, and it must be reevaluated repeatedly (as part of $p(\boldsymbol{\lambda})$) when resampling via RIS. Fortunately, we can select target PDF $\hat{p}(\boldsymbol{\lambda})$ to cancel terms in $p(\boldsymbol{\lambda})$, avoiding explicit evaluation of T in $p(\boldsymbol{\lambda})$. But when spatiotemporally reusing samples, this cancellation is no longer possible; thus, we must replace delta tracking with another sampling method, as discussed in Section 5.4.

A random walk up to (a user-defined maximum of) K steps generates up to K scattering paths with $0 < k \leq K$ and K emission paths with $0 \leq k < K$, as shown in Figure 5.3. But a random walk may terminate early, if it exits the media prior to the K^{th} scattering event. Let n be the total number of paths generated by a random walk (i.e., $n \leq 2K$). If sampling one of these n paths uniformly, the joint sampling PDF $\bar{p}(\boldsymbol{\lambda})$ can be written relative to the PDF of the random walk $p(\boldsymbol{\lambda})$ as

$$\bar{p}(\boldsymbol{\lambda}) = \frac{1}{n}\, p(\boldsymbol{\lambda})\ . \tag{5.16}$$

To generate more candidate paths, we can perform additional random walks starting from $\mathbf{x}_0$.

5.3.3 RIS Estimation of Volume Rendering

We generate paths using M random walks. Each random walk j produces n_j paths. Uniformly selecting one of the n_j paths as a sample for the path integral leads to high variance. Instead, we use RIS to select one path from each random walk to estimate the path integral, treating the n_j paths as stratified samples with source PDF $\bar{p}(\boldsymbol{\lambda}_j^i)$ for each path $i \in \{1, ..., n_j\}$ and using a target PDF $\hat{p}(\boldsymbol{\lambda}_j^i)$. If $M = 1$, then this RIS estimator can be written as

$$\langle L(\mathbf{x}_0, \boldsymbol{\omega}_o) \rangle_{\text{ris}}^{1,1} = E_{\hat{p}}(\boldsymbol{\lambda}_j^r) \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{\hat{p}(\boldsymbol{\lambda}_j^i)}{\bar{p}(\boldsymbol{\lambda}_j^i)} = E_{\hat{p}}(\boldsymbol{\lambda}_j^r) \sum_{i=1}^{n_j} \frac{\hat{p}(\boldsymbol{\lambda}_j^i)}{p(\boldsymbol{\lambda}_j^i)} \tag{5.17}$$

where $E_{\hat{p}}(\boldsymbol{\lambda}_j^r) = F(\boldsymbol{\lambda}_j^r)/\hat{p}(\boldsymbol{\lambda}_j^r)$ and $\boldsymbol{\lambda}_j^r$ represents the selected path. Notice the $1/n_j$ factor cancels the same value inside $\bar{p}(\boldsymbol{\lambda}_j^i)$. Appendix A has a more rigorous derivation. Given M random walks, we again resample to select one of the M paths $\boldsymbol{\lambda}_1^r, ..., \boldsymbol{\lambda}_M^r$ (note index r varies with j). As each candidate path $\boldsymbol{\lambda}_j^r$ comes from a prior RIS step, they must be weighted appropriately (by the running sum from the prior RIS round). Thus, the RIS estimator to select one path out of all M random walks is

$$\langle L(\mathbf{x}_0, \boldsymbol{\omega}_o) \rangle_{\text{ris}}^{1,M} = E_{\hat{p}}(\boldsymbol{\lambda}_r) \left(\frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{n_j} w(\boldsymbol{\lambda}_j^i) \right) . \tag{5.18}$$

We simplify the notation to $\boldsymbol{\lambda}_r$ (instead of $\boldsymbol{\lambda}_{r'}^r$) to represent the final path sample from this round of RIS. Here both $E_{\hat{p}}(\boldsymbol{\lambda}_r) = F(\boldsymbol{\lambda}_r)/\hat{p}(\boldsymbol{\lambda}_r)$ and $w(\boldsymbol{\lambda}_j^i) = \hat{p}(\boldsymbol{\lambda}_j^i)/p(\boldsymbol{\lambda}_j^i)$ depend on chosen target PDF $\hat{p}(\boldsymbol{\lambda})$.

Ideally, $\hat{p}(\boldsymbol{\lambda})$ closely matches $F(\boldsymbol{\lambda})$ but is cheaper to compute, as $\hat{p}(\boldsymbol{\lambda})$ gets evaluated for each candidate path sample. Therefore, we use $\tilde{F}(\boldsymbol{\lambda})$, a cheaper approximation of $F(\boldsymbol{\lambda})$, for paths with next event estimation

$$\hat{p}(\boldsymbol{\lambda}) = \begin{cases} \tilde{F}(\boldsymbol{\lambda}) & \text{if scattering path,} \\ F(\boldsymbol{\lambda}) & \text{if emission path.} \end{cases} \tag{5.19}$$

This approximation comes from simply using a cheaper transmittance estimate $\tilde{T}$ for light samples (discussed in Section 5.5.1):

$$\tilde{F}(\boldsymbol{\lambda}) = \Gamma_s(\boldsymbol{\lambda}) \, \tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1}) \, G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1}) \, L(\mathbf{x}_{k+1} \to \mathbf{x}_k) \, . \tag{5.20}$$

This particular definition of $\hat{p}(\boldsymbol{\lambda})$ includes the same transmittance terms T as $p(\boldsymbol{\lambda})$. Thus, all T terms cancel when computing $w(\boldsymbol{\lambda}) = \hat{p}(\boldsymbol{\lambda})/p(\boldsymbol{\lambda})$, such that

$$w(\boldsymbol{\lambda}) = W_k(\boldsymbol{\lambda}) \prod_{i=1}^{k} \frac{\sigma_s(\mathbf{x}_i)}{\sigma_t(\mathbf{x}_i)} \, , \tag{5.21}$$

where

$$W_k(\boldsymbol{\lambda}) = \begin{cases} \frac{\rho_k\, \tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\, G_{k+1}\, L^s(\mathbf{x}_{k+1} \rightarrow \mathbf{x}_k)}{p_{\text{NEE}}(\boldsymbol{\omega}_k | \mathbf{x}_k)} & \text{if scattering path,} \\ \frac{\sigma_a(\mathbf{x}_{k+1})}{\sigma_t(\mathbf{x}_{k+1})}\, L_e^m(\mathbf{x}_{k+1} \rightarrow \mathbf{x}_k) & \text{if emission path.} \end{cases} \tag{5.22}$$

This particular choice for $\hat{p}(\boldsymbol{\lambda}_r)$ also simplifies the computation of the $E_{\hat{p}}(\boldsymbol{\lambda}_r) = \frac{F(\boldsymbol{\lambda}_r)}{\hat{p}(\boldsymbol{\lambda}_r)}$ term, such that

$$E_{\hat{p}}(\boldsymbol{\lambda}_r) = \begin{cases} \frac{T(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})}{\tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})} & \text{if scattering path,} \\ 1 & \text{if emission path.} \end{cases} \tag{5.23}$$

Note the only remaining T term is needed to compute direct illumination for the one chosen path sample $\boldsymbol{\lambda}_r$ in $E_{\hat{p}}(\boldsymbol{\lambda}_r)$. This T term appears in no PDFs, only final shading, so we can afford unbiased estimates or even analytical methods. We discuss our choices for evaluating and estimating T in Section 5.5.1.

All source PDFs, however, cannot use stochastic estimation or biased approximation. While resampling allows arbitrarily *defining* $\hat{p}(\boldsymbol{\lambda})$ (including approximations), approximating source PDF $p(\boldsymbol{\lambda})$ after choosing sample $\boldsymbol{\lambda}$ introduces estimation error. By carefully choosing $\hat{p}(\boldsymbol{\lambda})$, we avoid expensive T terms in our PDFs and allows building an efficient RIS estimator for volume rendering with multiple scattering and volumetric emission.

This RIS estimator can be performed in a streaming manner using weighted reservoir sampling [121], such that only the one selected sample per pixel is stored, instead of explicitly storing all M candidate path samples.

5.4 Spatiotemporal Reuse

Quality of the RIS estimator in Equation 5.18 depends on the candidate sample count M. By reusing a pixel's candidate samples when evaluating neighbor pixels, we can substantially increase the effective per-pixel candidate sample count with minimal overhead. Similar to the direct illumination sampling in ReSTIR [7], we leverage streaming RIS and screen-space spatiotemporal reuse, storing intermediates in per-pixel reservoirs.

We generate M candidate samples $\boldsymbol{\lambda}_1^r, ..., \boldsymbol{\lambda}_M^r$ for each pixel. Each pixel selects one candidate $\boldsymbol{\lambda}_r$ using RIS and weighted reservoir sampling. We then consider the samples selected in nearby reservoirs and from the prior frame. Combining these reservoirs, we pick one sample per pixel to evaluate.

While our reuse follows the pattern of Bitterli et al. [7], key changes are needed to reuse volumetric path samples. Generating candidate paths for spatiotemporal reuse requires sampling a closed-form PDF, which requires updating the candidate generation process in Section 5.3.3, as we describe in Section 5.4.1. We discuss reusing paths between reservoirs in Section 5.4.2; unlike reusing direct light samples in ReSTIR, paths can be mapped between reservoirs in different ways with varying trade-offs. We introduce changes for spatial reuse in Section 5.4.3, refining the transmittance estimation and removing bias via different MIS weighting. Finally, in Section 5.4.4 we introduce a new stochastic reprojection for temporally reusing volumetric path samples, as surface motion vectors fail in volumes. Combined spatiotemporal reuse dramatically increases effective sample count, giving better quality than either reuse alone, as shown in Figure 5.4.

Importantly, our work combines transmittance estimates of varying quality, as we need a closed-form PDF for distance sampling, an efficient way to resample transmittance spatiotemporally, and unbiased transmittance for final shading (see Sections 5.4.1 and 5.4.3). We exploit resampling to iteratively refine transmittance, doing more expensive computations at lower frequency (see Figure 5.5).

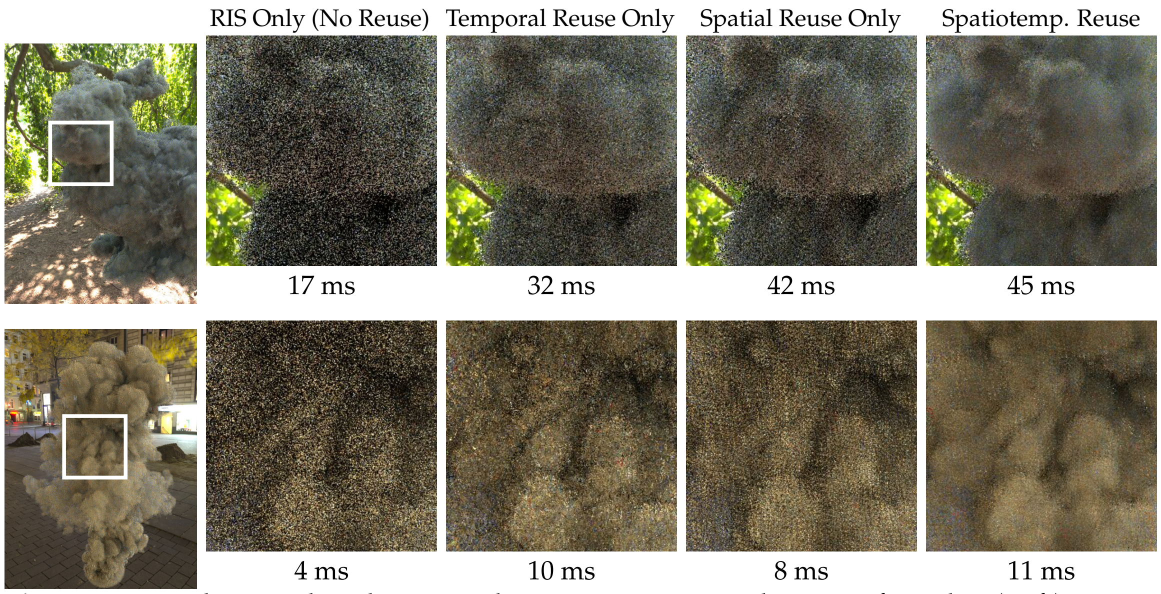


Figure 5.4: Both spatial and temporal reuse improve quality significantly. (Left) scenes with spatiotemporal reuse, and (right) insets comparing quality and performance without reuse, with only spatial or temporal reuse alone, and with spatiotemporal reuse. (Top) the *Bunny Cloud* scene uses a quickly rotating environment map and (bottom) the *Plume* scene has dynamic volume data.

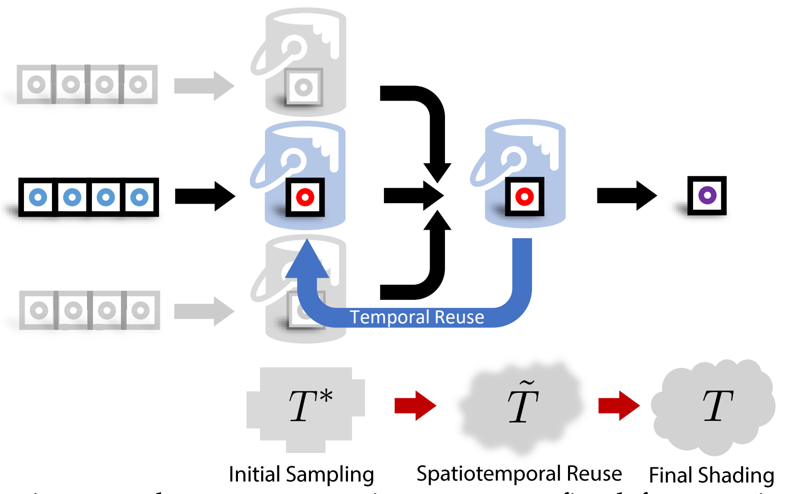


Figure 5.5: During sample reuse, transmittance gets refined from an initial piecewise constant approximation T^* to trilinear interpolation for ray marching $\tilde{T}$ to an analytical evaluation for shading T. But we always compute transmittance for NEE via ray marching (except final shading). Buckets visualize reservoirs, with initial candidates marked in blue, reservoir samples in red, and final shaded samples in purple.

5.4.1 Generating Candidate Samples for Reuse

When reusing samples $\boldsymbol{\lambda}$ from neighbor pixels or prior frames, we must explicitly compute (i.e., *resample*) the target PDF $\hat{p}(\boldsymbol{\lambda})$ at the current pixel and frame. Thus, using a target $\hat{p}(\boldsymbol{\lambda})$ containing expensive transmittance terms T (as in Section 5.3) makes spatiotemporal reuse computationally infeasible. But not including T in $\hat{p}(\boldsymbol{\lambda})$ means cancellation will not occur (in Equation 5.21) while computing $w(\boldsymbol{\lambda}) = \hat{p}(\boldsymbol{\lambda})/p(\boldsymbol{\lambda})$, requiring explicit transmittance computation for each candidate path (in $p(\boldsymbol{\lambda})$). To avoid this expense, we avoid using T terms in both $p(\boldsymbol{\lambda})$ and $\hat{p}(\boldsymbol{\lambda})$, replacing delta tracking (during candidate path generation) with an alternative distance sampling method with a closed-form PDF that is cheap to evaluate.

We use regular tracking [81]. Regular tracking may not outperform delta tracking, but it can be accelerated using a piecewise-constant approximation of the volume. For voxelized volumes, all points $\mathbf{x}$ within voxel v with constant density $\sigma^*_{t,v}$ get $\sigma^*_t(\mathbf{x}) = \sigma^*_{t,v}$. Let T^* denote transmittance between two points in this piecewise-constant volume. The PDF for regular tracking with piecewise-constant volume is then

$$p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1}) = T^*(\mathbf{x}_{i-1}, \mathbf{x}_i)\, \sigma^*_t(\mathbf{x}_i)\ . \tag{5.24}$$

Here, the transmittance term can be written as

$$T^*(\mathbf{x}_{i-1}, \mathbf{x}_i) = \prod_v e^{-\sigma^*_{t,v} d_{i,v}} \ , \tag{5.25}$$

where $d_{i,v}$ is the length of line segment $\overline{\mathbf{x}_{i-1}\mathbf{x}_i}$ inside voxel v (thus, $d_{i,v} = 0$ for voxels that do not intersect $\overline{\mathbf{x}_{i-1}\mathbf{x}_i}$).

We use this piecewise-constant volume only for importance sampling, i.e., to generate candidates $\boldsymbol{\lambda}^i_j$ and evaluate their PDFs $p(\boldsymbol{\lambda}^i_j)$ and $\hat{p}(\boldsymbol{\lambda}^i_j)$. When computing final path throughput, $F(\boldsymbol{\lambda}_r)$, for the one chosen sample $\boldsymbol{\lambda}_r$ per pixel, we use the more expensive, unbiased transmittance function T and density σ_t.

Care is required when generating candidates from the piecewise-constant volume. If σ_t is non-zero *anywhere* inside voxel i, $\sigma^*_{t,i}$ for the voxel *must* be non-zero to avoid bias. For example, we cannot trilinearly interpolate a voxel grid for T and use nearest sampling for T^*. Nearest sampling can return zero some places where trilinear sampling gives non-zero density, which would introduce bias. To avoid this, we use nearest sampling for T^*, except in zero-density voxels where we return the average density of their neighbors.

For further acceleration, we can use a lower resolution piecewise-constant volume for path generation. Our results (Section 5.5.1) show that defining this piecewise-constant volume at lower resolution than the original volume substantially improves performance with only minor quality impacts.

5.4.2 Path Reuse

To reuse paths we must create a path $\boldsymbol{\lambda}$ with vertices $\mathbf{x}_0, \ldots, \mathbf{x}_{k+1}$ in pixel q based on a path $\boldsymbol{\lambda}'$ from a different pixel q' with vertices $\mathbf{x}'_0, \ldots, \mathbf{x}'_{k+1}$. As both $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ start at the same camera position, we get $\mathbf{x}_0 = \mathbf{x}'_0$. However, the same is not true for the other vertices. The pixels may have different primary ray directions $\boldsymbol{\omega}_0 \neq \boldsymbol{\omega}'_0$, so the next vertex $\mathbf{x}_1 = \mathbf{x}_0 + z_1\boldsymbol{\omega}_0$ must be different as well (i.e., $\mathbf{x}_1 \neq \mathbf{x}'_1$). We can, however, use the same distance along the primary ray for both paths, such that $z_1 = z'_1$.

For the following vertices, we consider two options (Figure 5.6):

- **Vertex reuse** by simply setting $\mathbf{x}_i = \mathbf{x}'_i$ for $i \geq 2$, or
- **Direction reuse** by taking $\boldsymbol{\omega}_i = \boldsymbol{\omega}'_i$ and $z_i = z'_i$.

Vertex reuse reduces computation, as we need not recompute $T^*(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$ for $i \geq 1$. However, it includes an unbounded geometry term $G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = 1/\left|\mathbf{x}_2 - \mathbf{x}_1\right|^2$ that

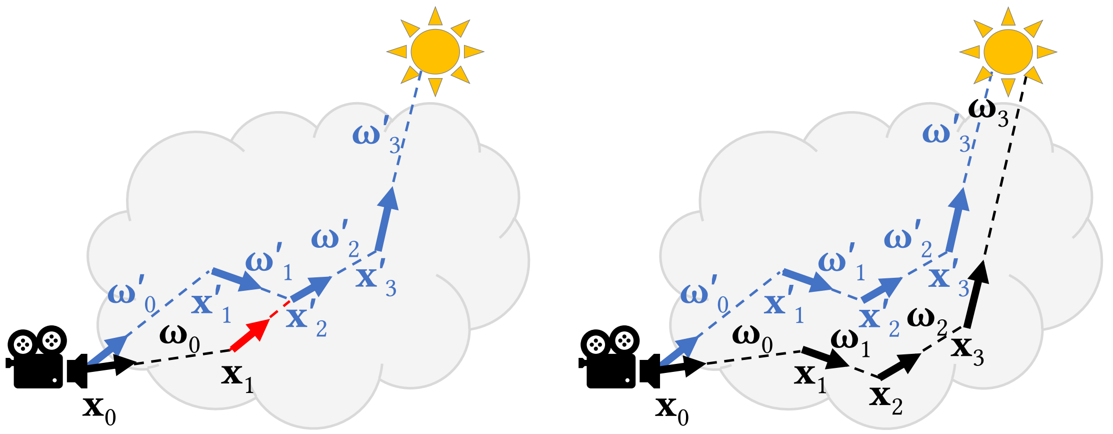


Figure 5.6: Path λ (black) is created from a neighbor pixel's path λ' (blue). Vertex $\mathbf{x}_1$ is the same distance along primary ray ω_0 (as $\mathbf{x}'_1$ along ω'_0). Vertex reuse connects $\mathbf{x}_1$ to $\mathbf{x}'_2$ (red segment) to form the rest of λ. Direction reuse takes $\omega_i = \omega'_i$ and $z_i = z'_i$ along the remaining path.

introduces fireflies. In Bitterli et al. [7], singularities occur around corners and edges, but in volumes they can occur anywhere.

Direction reuse must compute $T^*(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$, but avoids these artifacts. It is also possible to combine both approaches by reusing directions for a desired number of scattering events then switching to vertex reuse; this reduces the probability of fireflies and bounds the cost. Our experiments show reduced noise for a slight cost increase with direction reuse, so results in the chapter all rely on direction reuse. While long very paths may be initially generated, they are unlikely to be selected and reused via RIS as they carry less energy. This helps bounds the average cost of direction reuse.

As for the last vertex $\mathbf{x}'_{k+1}$, if on a light surface or in emissive media, we take $\mathbf{x}_{k+1} = \mathbf{x}'_{k+1}$. If our reused path samples the environment map, we take $\omega_k = \omega'_k$.

Vertex reuse in the presence of surfaces is straightforward, by simply adding surface vertices to the path. For direction reuse, if $\mathbf{x}'_i$ lies on a surface, we put $\mathbf{x}_i$ onto the closest surface along the ray starting at $\mathbf{x}_{i-1}$ with direction ω_{i-1}. Hence, if the scene does not contain a volume, only reflection directions are reused.

5.4.3 Spatial Reuse

After generating M per-pixel candidates, each pixel q's reservoir has selected a path λ_q (i.e., λ_r for each q). Next, we spatially reuse from neighbor pixel reservoirs, using RIS to combine the neighbor reservoirs with the current pixel's. For each neighbor q', this involves computing correction factor $w_{q'\to q}$ (per Equation 5.5), using

$$w_{q'\to q} = \frac{\hat{p}_q(\boldsymbol{\lambda}_{q'})}{\hat{p}_{q'}(\boldsymbol{\lambda}_{q'})} w_{q'}^{\text{sum}}, \qquad \text{where} \qquad w_{q'}^{\text{sum}} = \sum_{j=1}^{M} \sum_{i=1}^{n_j} \frac{\hat{p}_{q'}(\boldsymbol{\lambda}_j^i)}{p_{q'}(\boldsymbol{\lambda}_j^i)}\,. \tag{5.26}$$

Here, the same path sample $\boldsymbol{\lambda}_{q'}$ contains different vertices in pixels q and q' due to the reuse of z_1 along different rays and later direction reuse. Therefore, when computing $\hat{p}_q/\hat{p}_{q'}$ in $w_{q'\to q}$, not all transmittance terms in the PDFs cancel, as they are evaluated for different pixels (none of them cancel with direction reuse and only some of them cancel with vertex reuse).

As noted in Section 5.4.1, as a byproduct of distance sampling we used a piecewise-constant volume to compute transmittance T^*. However, when recomputing transmittance values during reuse, there is no computational need for such simplification (cancellation of terms generally cannot happen between neighboring pixels). Plus, the piecewise-constant sampling enlarges non-zero density regions and leads to suboptimal sampling quality.

Instead of keeping the lower quality transmittance estimate T^*, during resampling we can update target function $\hat{p}_q$ to use higher quality transmittance than the input $\hat{p}_{q'}$ values. For resampling $\hat{p}_q$ during reuse, we compute a new transmittance estimate, $\tilde{T}$, using ray-marching with trilinearly filtered densities to improve subsequent sample quality. Because this (biased) ray marching is only used during importance sampling, and not for final throughput in $F(\boldsymbol{\lambda})$, it does not bias the rendering. An additional advantage of ray marching is the ability to tune step size, depending on our resampling budget.

Thus, $\hat{p}_q \neq \hat{p}_{q'}$ even for $q = q'$, simply because we use an updated target function for $\hat{p}_q$ (with $\tilde{T}$ instead of T^*) for $\hat{p}_q$. This improved transmittance estimate behaves similar to *visibility reuse* from Bitterli et al. [7].

We must also consider that some valid path samples $\boldsymbol{\lambda}$ for pixel q may never be sampled by a neighboring pixel q', i.e., $p_{q'}(\boldsymbol{\lambda})$ may be zero for some $\boldsymbol{\lambda}$ with non-zero $p_q(\boldsymbol{\lambda})$. Simply ignoring this introduces sampling bias, excessively darkening the results. Bitterli et al. [7] correct this via stochastic MIS weighting (i.e., Equation 5.7). Although faster than the deterministic MIS (Equation 5.8) weighting introduced by Talbot [8], we found stochastic MIS excessively noisy in volumes, as shown in Figure 5.7.

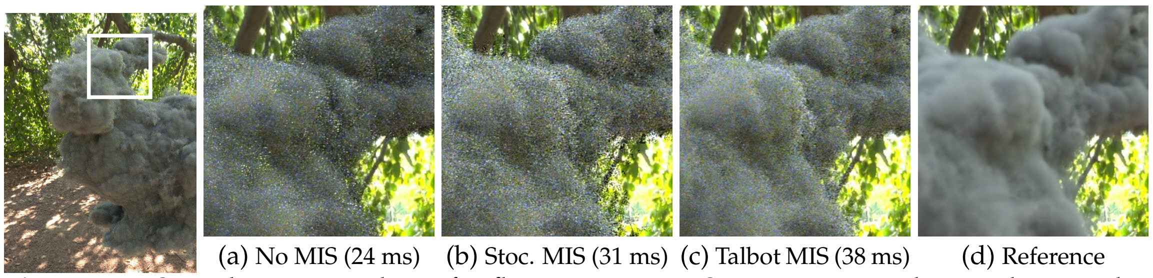


Figure 5.7: Sample reuse without fireflies requires MIS to appropriately weight samples. Bitterli et al. [7] introduced a cheaper $O(N)$ stochastic MIS, though for our volume formulation the more expensive Talbot MIS [8] works better.

Heuristically rejecting spatial neighbors based on features like surface normal or depth is effective for reducing the noise on surfaces [150], but for volumes such features are stochastic, making heuristics-based rejection challenging. Instead, we use Talbot MIS for spatial reuse; while it has quadratic cost, this is acceptable when using a small number of spatial neighbors.

5.4.4 Temporal Reuse

Temporal reuse significantly improves sample quality by incorporating knowledge from prior frames. The challenge for such reuse is finding relevant samples by *temporally reprojecting* prior frames, including changes from camera motion and volumetric deformation.

But temporal reprojection is ill-defined for volume rendering. The media in any pixel may move in many directions, so no single "correct" motion vector can tell us what prior-frame data should be reused.

We approach the problem probabilistically. With temporal reprojection we seek motion vectors that select, with high probability, prior frame reservoirs containing useful data. For example, if most media in a pixel has one motion vector, reusing a reservoir corresponding to that motion likely reduces variance best (e.g., preferentially sampling motion from denser media in a pixel instead of following a closer, wispy cloud's motion).

To that end, we use the motion vector at $\mathbf{x}_1$, the first vertex on pixel q's selected path $\boldsymbol{\lambda}_q$ (prior to spatial reuse). To compute the motion vector, we treat $\mathbf{x}_1$ as a particle such that its previous frame's position is determined by the velocity field of the volume. This randomizes the choice of motion vector, allowing any visible media to (potentially) contribute motion, using the target PDF $\hat{p}(z_1|\mathbf{x}_0, \boldsymbol{\omega}_0)$. Media with higher $\hat{p}(z_1|\mathbf{x}_0, \boldsymbol{\omega}_0)$ has a higher probability to provide the motion vector, which is reasonable as it contributes more pixel radiance.

Figure 5.8 shows examples of camera animation and volumetric deformation with and without temporal reprojection. Notice that temporal reprojection can help reduce the noise substantially.

An important limitation of this temporal reuse and reprojection is it remains unbiased only for static volumes and camera. Under camera motion or volume deformation, the chosen temporal reservoir depends on z_1 (i.e., the first scatter event). This turns the target PDFs in RIS into conditional PDFs (conditioned on z_1), introducing a slight bias during reuse if treated as a marginalized PDF.

Bias increases with larger camera motion or volume deformation (see Figure 5.9). But the bias is generally hard to perceive. Figure 5.9 shows examples with fast camera motion and large volume deformation, but only slight darkening/brightening happens. Lowering Q, the temporal limiting factor (see Section 5.2.2), reduces bias, and the bias disappears entirely a few frames after motion ends.

Note that dynamic lighting does not add bias. Instead, sudden lighting changes effectively lower the PDF for temporal candidates, increasing variance near lighting discontinuities.

5.5 Implementation Details

The above volumetric sampling techniques can be implemented in various ways. In this section, we provide the details of our prototype.

Our implementation has four passes, similar to the flow in Figure 5.2. First, we generate initial candidate paths for each pixel and pick one, via RIS, to share with neighbors. Second, we reproject to find a temporal neighbor for reuse and again resample. Third, we perform spatial resampling. Finally, we evaluate the selected path sample for shading. We visualize our pipeline in Figure 5.5. Our reservoir stores full paths as a list of $(z_i, \boldsymbol{\omega}_i)$ tuples. Memory costs are bounded by the allowed number of scattering events, K. But supporting infinite bounces is possible by switching to vertex reuse after a few bounces and caching incident radiance of the remaining path.

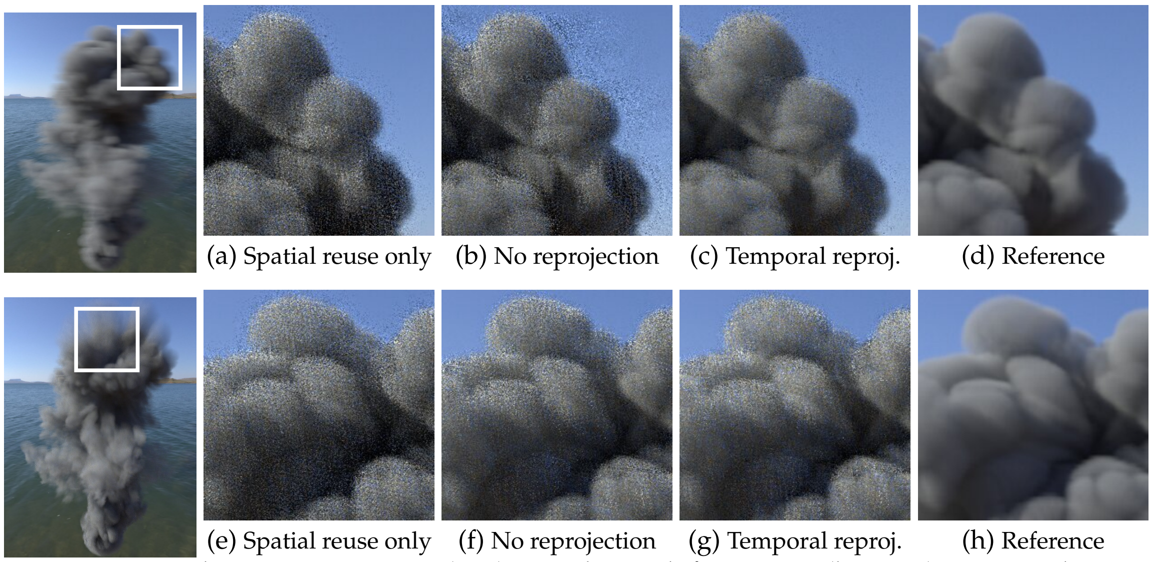


Figure 5.8: Under camera motion (top) or volume deformation (bottom), temporal reprojection helps identify good samples for reuse. At left, we show references using motion blur to illustrate the magnitude of per-frame motion. At right, we show insets from a single frame without motion blur. (a,e) Only spatial reuse within the current frame. (b,f) Without temporal reprojection we reuse from inappropriate prior frame locations, causing halos and masking the noise reduction from temporal reuse. (c,g) Our novel temporal reprojection reduces the haloing and generally reduces noise.

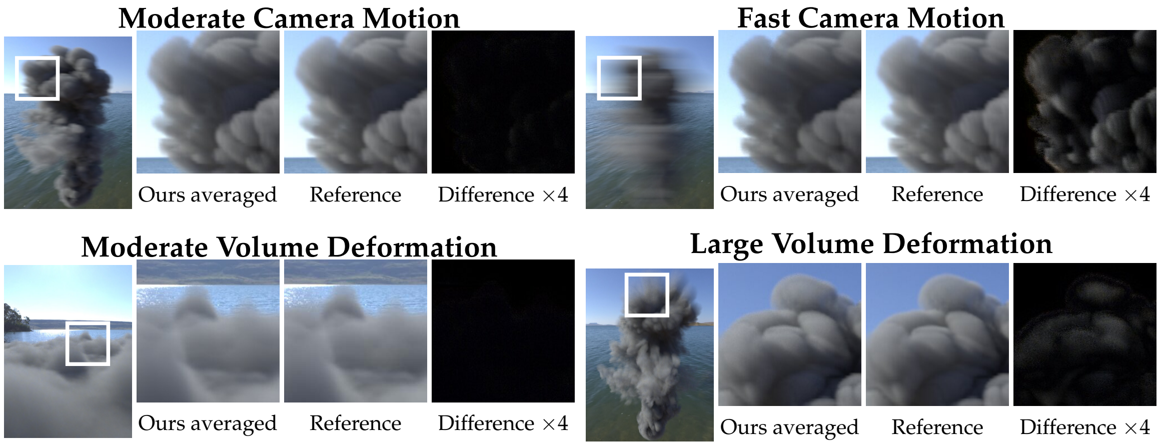


Figure 5.9: Bias in temporal reuse with (top) camera motion and (bottom) volume deformation, comparing the results of our method averaged over 256 recomputations of the same frame (to produce nearly-converged images) to reference images. (Left) with moderate motion/deformation bias is imperceptible, but (right) faster camera motion or larger volume deformation increases this bias. Note that we use a slowly deforming fog as the example for moderate volume deformation, which is different from other images. The full images on the left are rendered with motion blur to illustrate the magnitude of the camera motion or volume deformation. The insets do not include motion blur.

5.5.1 Optimizing Transmittance Computation

Transmittance plays a vital role in volumetric resampling, as it contributes significant cost to target function $\hat{p}$ and must be evaluated between every two path vertices. Prior work [7] notes resampling efficiency is maximized when choosing a target function $\hat{p}$ that closely approximates integrand f but is much cheaper to evaluate.

When computing transmittance $\tilde{T}$ in $\hat{p}$ for spatiotemporal reuse, we ray march a coarser volume (i.e., Mip 1, the original volume downsampled by half). Our step size for ray marching is the diagonal of a Mip 1 voxel. We lossily compress the downsampled volume with DirectX's BC4 block compression format, which compresses density values to 4 bits, giving a total 64:1 compression from the original; this greatly reduces sampling bandwidth, improving performance.

Directly rendering such volumes causes overblurring, but we use it just for importance sampling. Figure 5.10 compares ray marching our downsampled volume with analytical transmittance computations in the original volume. The downsampled volume slightly reduces sampling quality, but significantly improves performance. But coarsening can go too far; Figure 5.10c uses a Mip 3 volume for importance sampling. While cost drops further, sampling quality decreases significantly.

However, Figure 5.11 shows initial candidate paths can ray march a Mip 2 volume to estimate the transmittance on NEE segments (i.e., for volumetric shadows) without affecting sampling quality. This shows the benefit of incrementally injecting higher quality

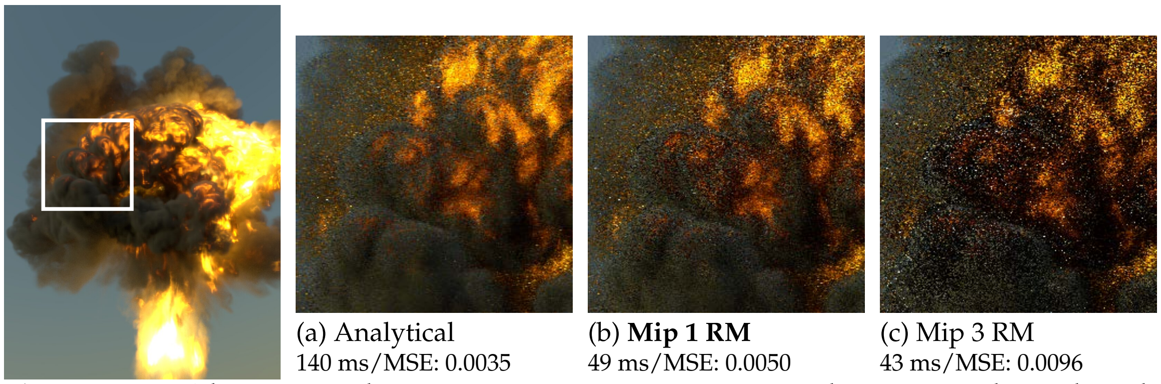


Figure 5.10: When resampling, approximating transmittance by ray marching though coarser volumes (Mip 1 RM) greatly lowers cost, in exchange for a little noise (compared to analytical transmittance). But lowering resolution too far (e.g., Mip 3) adds more noise without much speedup.

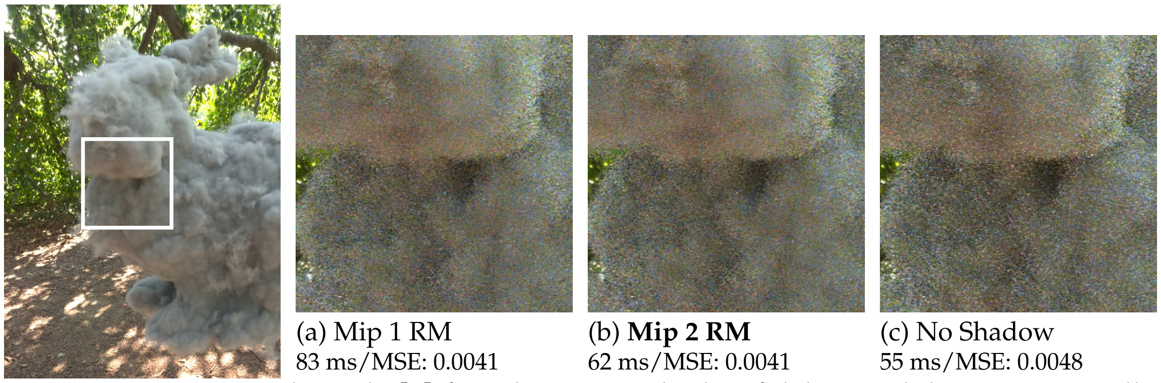


Figure 5.11: As Bitterli et al. [7] found, injecting higher fidelity visibility incrementally during resampling improves quality without the cost to compute it everywhere. Using (c) no transmittance for NEE segments increases noise due to poorer sample quality. Adding transmittance reduces noise, but (b) even very crude approximations (e.g., ray marching a $1/4^3$ sized volume) provide most of the benefits.

transmittance into target function $\hat{p}$ over multiple rounds of RIS, rather than always using the highest quality.

We also sample distances from downsampled volumes when generating initial path candidates. Figure 5.12 shows that mixing Mip 1 for sampling primary path segments and Mip 2 volume for indirect path segments yields similar quality as analytical regular tracking in the original volume, but with much higher performance. Again, coarsening too far significantly skews the sampling distribution, reducing quality (see Figure 5.12c).

In the integrand F, we analytically compute transmittance T by traversing the voxels using piecewise-trilinear regular tracking [85], giving a closed-form, unbiased transmit-

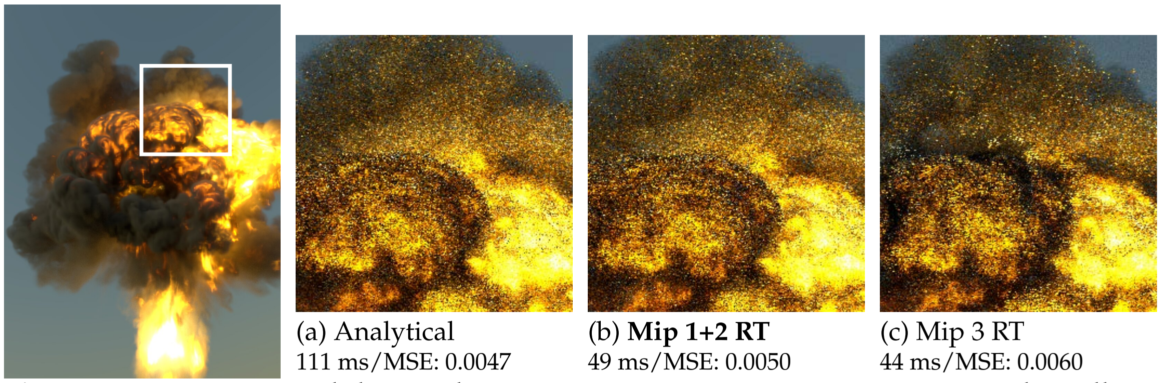


Figure 5.12: During candidate path generation, computing transmittance analytically is costly. We use regular tracking (RT) through coarser volumes to reduce cost with little impact to sample quality (using Mip 1 for primary and Mip 2 for indirect rays). Coarsening too far (e.g., Mip 3) noticeably reduces quality without much performance win.

tance. This traversal is expensive, fetching 8 density values per step to evaluate a cubic polynomial. However, we only do this for one path—the one selected for final shading. Should such a closed form solution be infeasible, we can switch to ratio tracking [6] for this final transmittance estimate; a large majorant should be used to minimize noise by forcing a smaller average step.

5.5.2 Velocity Resampling

Our temporal reprojection (Section 5.4.4) uses the motion vector of vertex $\mathbf{x}_1$ on each pixel's selected path $\boldsymbol{\lambda}_q$. Generally this works well, but near volume silhouettes, $\boldsymbol{\lambda}_q$ may not have vertices in the media, placing $\mathbf{x}_1$ on the background. This often moves differently than the volume, giving a halo along edges (see Figure 5.13).

We address this with *velocity resampling*. For any $\mathbf{x}_1$ not in the volume, we generate a new distance z corresponding to a particle in the volume proportional to the free flight distance $p(z) = \sigma_t(\mathbf{x}_1')T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1')$ with $\mathbf{x}_1' = \mathbf{x}_0 + z\boldsymbol{\omega}_0$, and use the motion vector for this sample. As z must be sampled in the volume, $p(z)$ is unnormalized. We approximate distribution $p(z)$ by assigning each voxel a weight proportional to its average $p(z)$, importance sampling along the ray, and uniformly picking a point within the selected voxel.

This approach increases the probability of picking a point in the media, providing better temporal reprojection. This reduces noise and bias from suboptimal reprojections, as temporal reservoirs are more likely to provide relevant paths that reduce variance during reuse.

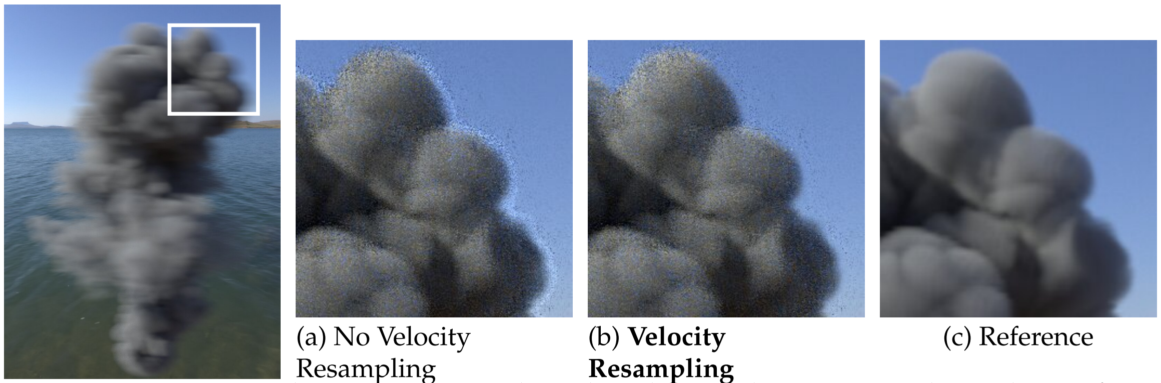

(a) No Velocity Resampling (b) **Velocity Resampling** (c) Reference

Figure 5.13: Temporal reprojection with and without velocity resampling, shown for a dynamic camera. (a) Overusing background motion along silhouettes causes brightening bias around the edge. (b) Velocity resampling fixes this.

5.5.3 Parameters of Spatiotemporal Reuse

Resampling has various parameters impacting quality and performance. First, we use $M = 4$ random walks to generate initial candidates. In scattering paths, this produces fewer light samples than the 32 light samples used by Bitterli et al. [7], but our path generation is more expensive, so we trade fewer initial samples for more spatiotemporal reuse, which maximizes the sampling efficiency in a given budget.

We typically use $Q = 4$ as the temporal limiting factor, controlling the maximum prior frame contribution. Larger Q accumulates more samples, but increases the chance of reusing stale temporal reservoirs, which can cause fireflies under large lighting changes (see Figure 5.14c).

However, in scenes combining volumes and surfaces under complex illumination, we use larger Q to accumulate more effective samples to reduce noise. The value chosen depends on which issue is more problematic. In scenes containing surfaces, we use $Q = 10$.

During spatial reuse, we use a low-discrepancy sequence to sample 3 random neighbors within a 10 pixel radius. This achieves a balance between correlation artifacts and error, as shown in Figure 5.15. We use direction reuse by default.

5.6 Results

We built our algorithm in the Falcor real-time rendering framework [151], and we used GVDB [152] to load and access VDB assets [153]. We captured results on an NVIDIA GeForce RTX 3090. Performance numbers include initial candidate generation, spatiotem-

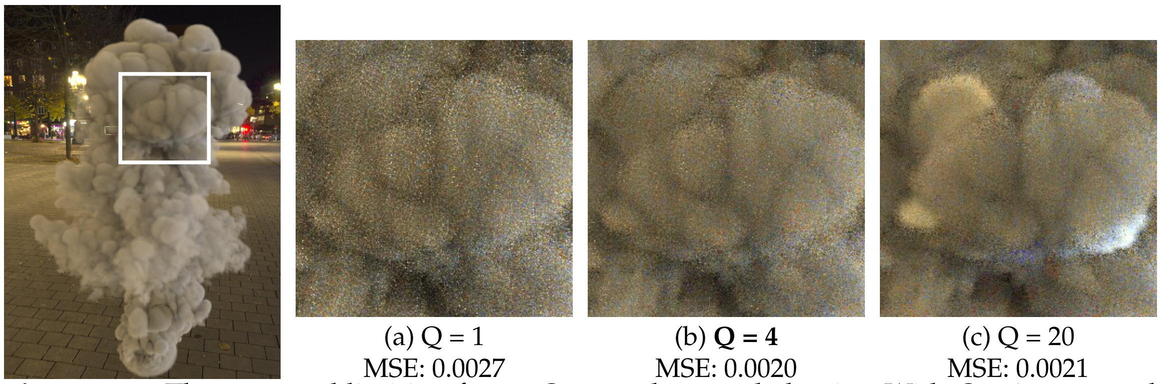


Figure 5.14: The temporal limiting factor Q controls reuse behavior. With $Q=1$, temporal samples get low relative weight. With $Q=20$, older frames have more aggregate impact; this reduces overall noise, but very old stale samples can get disproportionally weighted under quickly changing illumination, causing fireflies. We use $Q=4$.

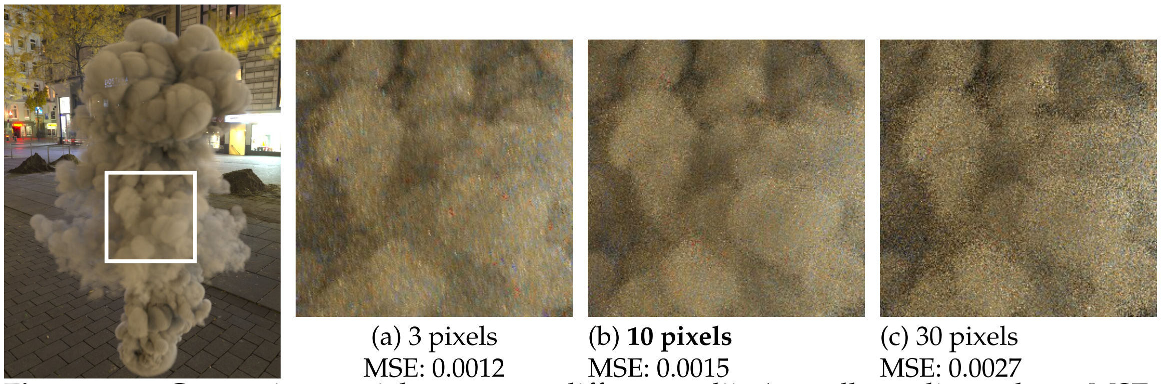


Figure 5.15: Comparing spatial reuse over different radii. A smaller radius reduces MSE, but correlation between nearby pixels becomes visually apparent. Increasing the radius reduces visual correlation, but also increases error. We use a 10 pixel radius, balancing these considerations.

poral reuse, and final shading. Scenes with surfaces use inline ray tracing to allow handling the surface visibility together with volumes in one compute shader.

We report error metrics and timing for 1920×1080 images, averaging over 256 frames (after a warmup) to smooth variations. We use HDR light probes and polygonal scenes with many emissive triangles to create realistic lighting environments.

Unless noted, all figures show static scenes and cameras and are fully unbiased. We do not leverage this to simplify, cancel, or otherwise reduce computation; thus, the timings are equivalent under animation. Scenes with dynamic cameras, volumes, and lighting are shown in our SIGGRAPH Asia 2021 paper's supplemental video.

We compare our results with a fast implementation of decomposition tracking [5] (to sample free flight distances) and residual ratio tracking [6] (for estimating transmittance in NEE). We call this our *baseline*. Both decomposition and residual ratio tracking use super-voxels [85] with $8\times$ the original voxel size to store local minimum, maximum, and average density values; these control volume densities as described by Novák et al. [6] and Kutz et al. [5]. For GVDB, we use $8\times8\times8$ voxel bricks [152]. This enables storing super-voxel density bounds in brick headers, allowing efficient fetches during VDB traversal. This maximizes our baseline's performance.

5.6.1 Single and Multiple Scattering Results

We show our method in six scenes: the *Bunny Cloud* in two lighting configurations (Figures 5.1 and 5.16), the *Disney Cloud* (Figure 5.16), an *Explosion* (Figure 5.17), an ani-

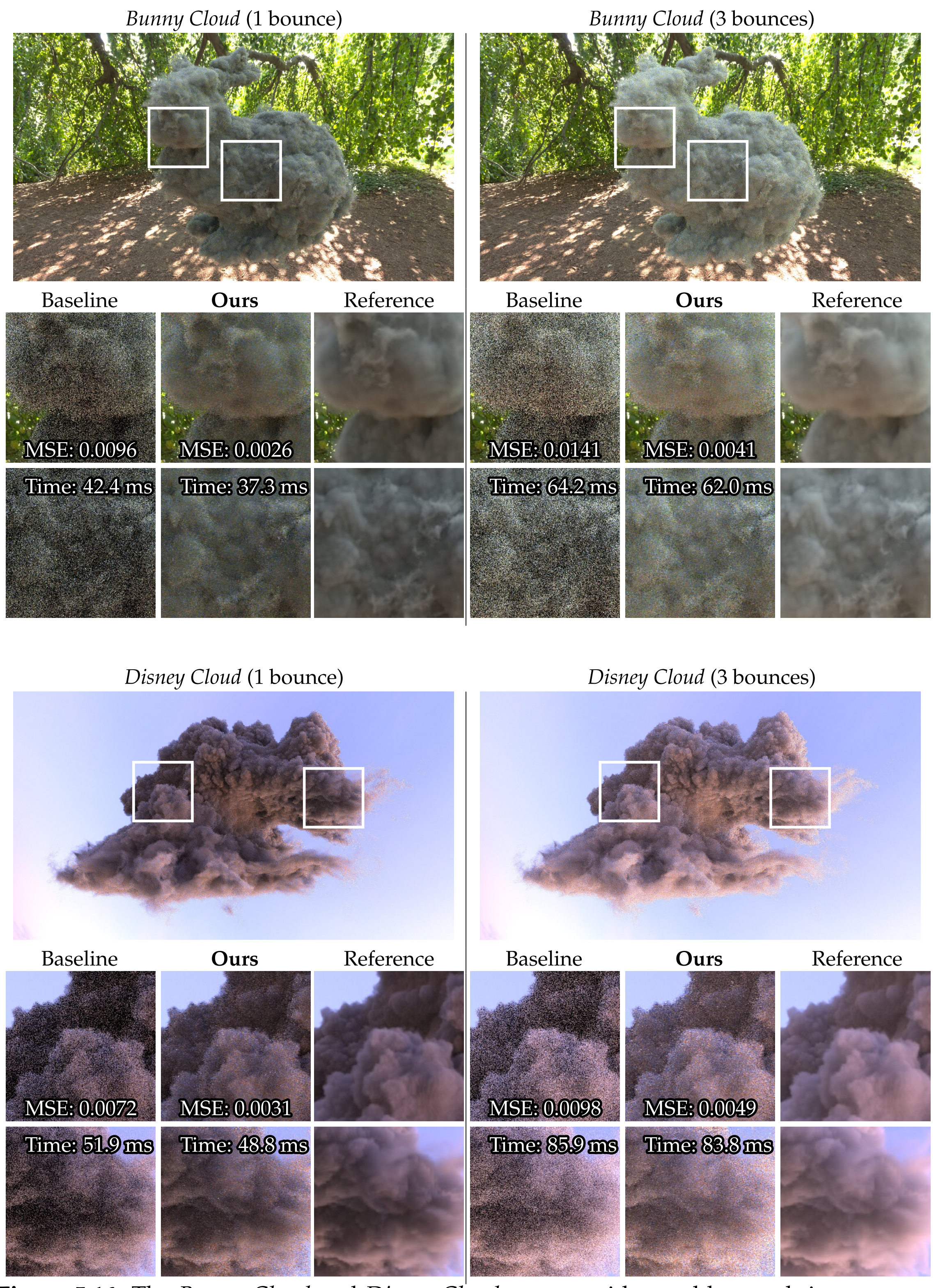


Figure 5.16: The *Bunny Cloud* and *Disney Cloud* scenes, with roughly equal-time comparisons between the baseline and our method.

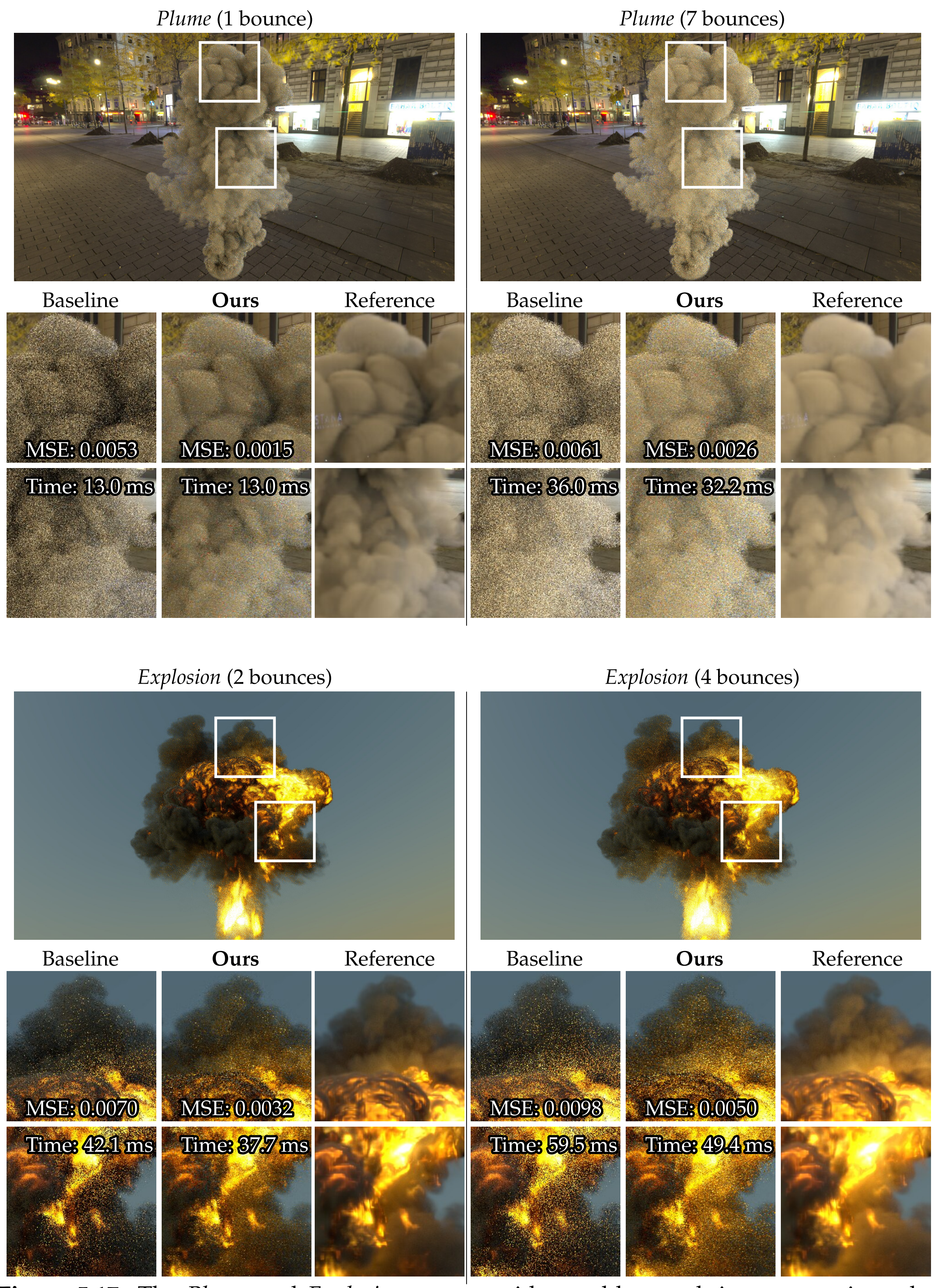


Figure 5.17: The *Plume* and *Explosion* scenes, with roughly equal-time comparisons between the baseline and our method.

mated *Plume* (Figure 5.17), the *Amazon Bistro* with a smoke plume (Figure 5.18), and the *Emerald Square* with fog (Figure 5.18). These cover various uses: emissive lights, complex environment lighting, volume self-emission, dynamic media, and volume-surface interactions. We use isotropic scattering ($g = 0$), unless otherwise stated. In all cases, our method significantly improves over the optimized baseline.

In real-time contexts, K provides an important performance knob, defining the allowable scattering events. Due to costs in dense voxel grids (e.g., the bunny), we limit evaluation in most scenes to $K = 3$. For $K > 3$, we apply Russian roulette after $\mathbf{x}_3$ to stochastically terminate a random walk according to the albedo of the scattering point. We do the same for terminating paths with the baseline. To produce equal-time comparison, we choose the number of samples per pixel (spp) for the baseline method to make the render time match our method. With each sample, the baseline method performs a random walk up to K scattering events and it performs NEE at each bounce, just like our initial path candidate generation. Note we only evaluate 1 spp per frame, in a Monte Carlo sense, but we generate and reuse many samples as part of importance sampling.

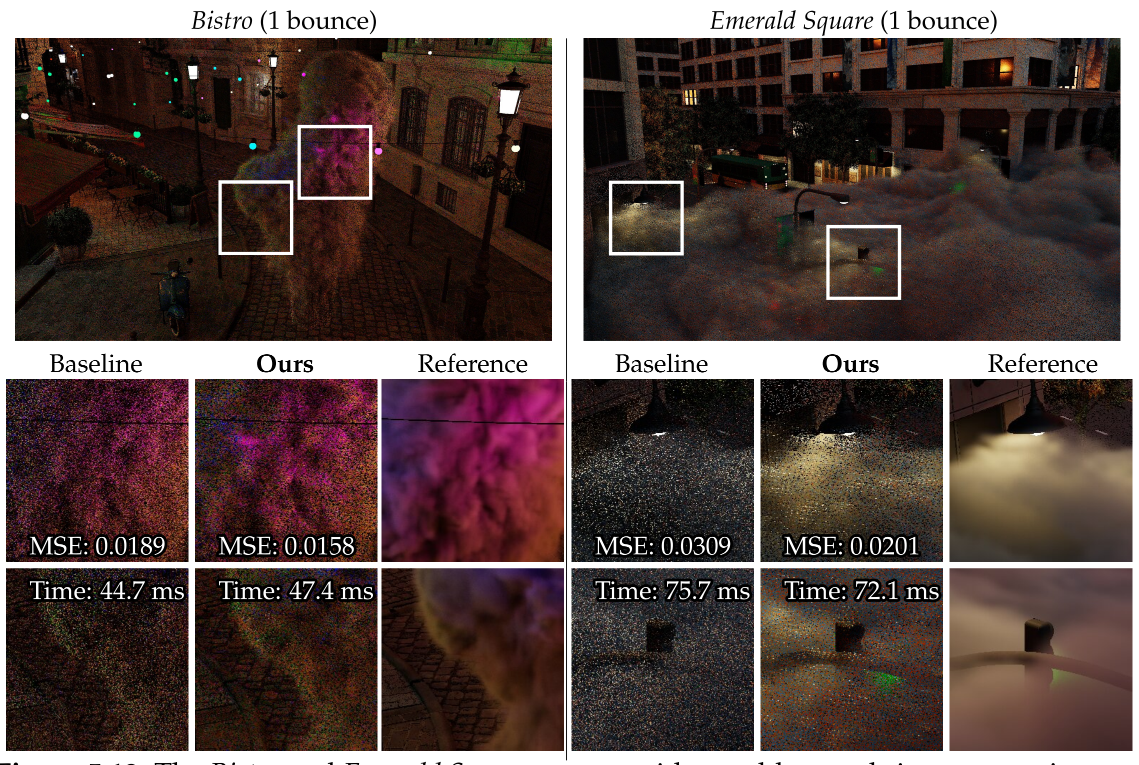


Figure 5.18: The *Bistro* and *Emerald Square* scenes, with roughly equal-time comparisons to the baseline.

Figures 5.16 and 5.17-top explicitly compare performance and quality in three scenes with environment lighting using varying number of maximum scattering events ($K = 1$, 3, and 7). We sample the environment proportional to texel intensity. In all cases, our approach significantly reduces error compared to equal-time baseline renderings.

In the *Explosion* scene with $K = 2$ and $K = 4$ scattering events, our method significantly reduces the emission sampling noise compared to the baseline. This provides better scattering quality (emissive voxels lighting other parts of the volume) as well.

For the *Plume* scene in Figure 5.17, our method significantly outperforms the baseline with both single scattering and multiple scattering up to 7.

We show results of volume single scattering from light sources and surface direct lighting in the *Bistro* with over 20k emissive triangles and *Emerald Square* with around 90k emissives (Figure 5.18), our work enables volumes to benefit from RIS and ReSTIR, similar to surfaces [7]. Here, the baseline uses a light BVH [3] for light sampling, while our method samples sources proportional to power. We show results with multiple scattering and multiple-bounce surface-volume interreflection in Figure 5.1. Note that this significantly increases render time, especially using multiple scattering. Profiler results reveal this stems from thread divergence between ray tracing and volume tracking.

In all scenes, our method gives lower MSE than the baseline with approximately equal or less time.

5.6.2 Participating Media with Different Densities

Our algorithm estimates transmittance using ray marching. Since path segments are longer in low density volumes, our algorithm accesses more data for the same model size.

In Figure 5.19, we scale the density of the *Disney Cloud* to $0.2\times$ and $3\times$ of the original for comparison. Our method's cost grows 47% from 83.8 ms to 123.1 ms. In comparison, the baseline method requires less time per sample, as decomposition tracking and residual ratio tracking take larger average flight distances between null collisions (due to smaller majorant).

At the lowest density, our method has slightly higher MSE than the baseline. Looking closely shows this is caused by color noise, a limitation of ReSTIR [7] caused by using the same scalar PDF to importance sample all color channels. We still achieve lower MSE for

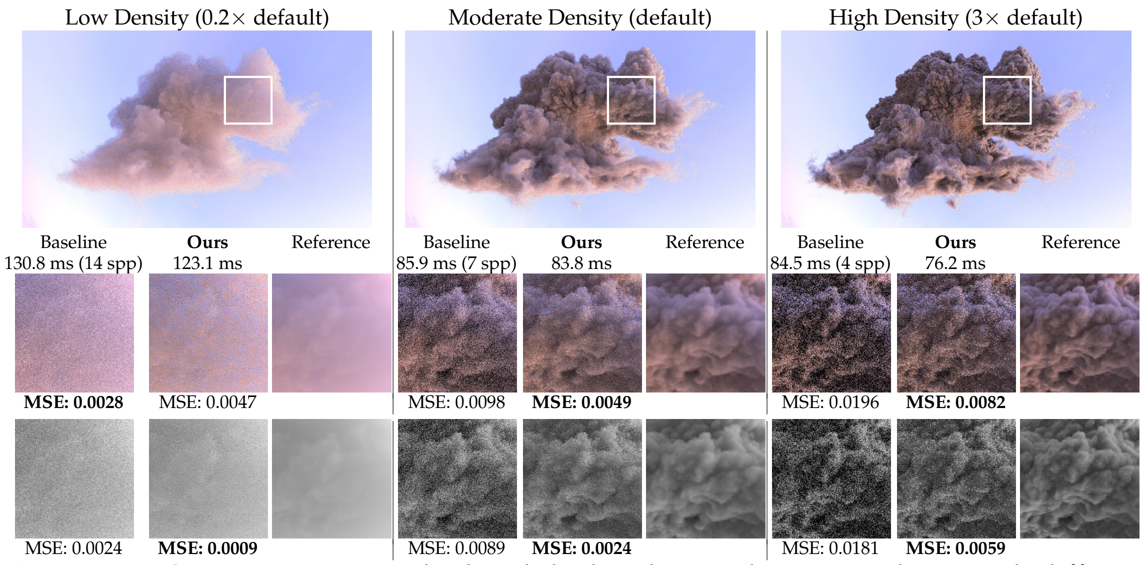


Figure 5.19: Comparing our method and the baseline in the same volume with different density multipliers. Notice that lower density makes the execution time of our method longer, while it reduces the time for each sample of baseline. We provide both color and monochromatic images to show the impact of color noise. Images shown with 3-bounce multiple scattering.

monochromatic images. At low densities, more background samples are produced. When the background color differs strongly from the scattered light, color noise is amplified. This is not bias; aggregating more frames reduces color noise (Figure 5.20) and converges to the reference. Note that increasing the initial candidate count M does not reducing color noise, as they all contribute our scalar target PDF, producing one integrand f whose chroma is randomized.

Conversely, increasing media density speeds our algorithm and makes each baseline sample more expensive. Note that we still have color noise, but its influence is smaller than the remaining variance in the overall integral.

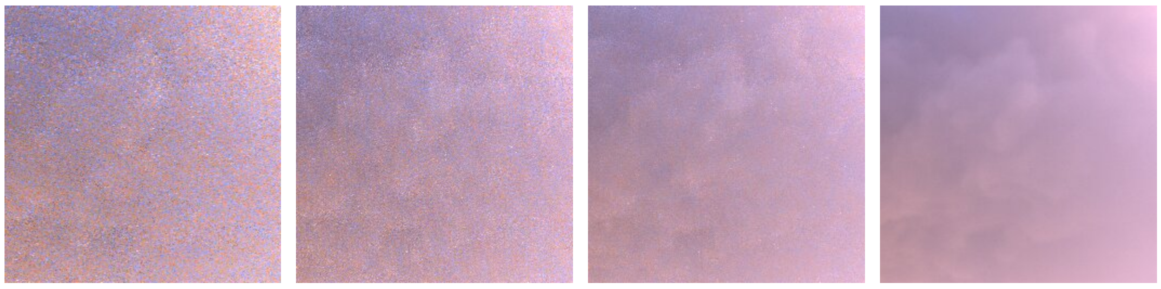


Figure 5.20: Color noise decreases when accumulating multiple frames.

5.6.3 Participating Media with High Anisotropy

To validate robustness with highly anisotropic phase functions, we change the Henyey-Greenstein asymmetry parameter g to 0.8 in the *Bunny Cloud*, causing strong forward-scattering (Figure 5.21 top). Our baseline, using only NEE, is outperformed by null scattering [9], which combines residual ratio tracking in NEE with the radiance of escaped ray (from free-flight sampling) using MIS; this is infeasible without null scattering. But our method still outperforms null scattering, despite only using NEE for light samples.

But null scattering does not always outperform decomposition tracking. For an isotropic phase function ($g = 0$) and complex lighting (Figure 5.21 bottom), the MIS in null scattering may not yield better quality and it adds cost to each sample. As null scattering generates fewer samples per pixel than our baseline for equal time, the sampling efficiency becomes lower. Here our method significantly outperforms both methods, despite having some color noise.

Note we could use MIS to sample candidate lights using the MIS weights for RIS [8]. Since our initial path generation operates with closed-form PDFs, we do not rely on the null scattering formulation to compute MIS weights. However, this adds overhead to candidate generation. To discover when MIS is most effective requires further investigation.

5.6.4 Longer Time Convergence

To compare how error evolves with time, we accumulate frames of both our method and baseline, comparing errors from 100 ms to 10 seconds (Figure 5.22). The *Bunny Cloud*, *Explosion*, and *Emerald Square* scenes are selected as representative of high albedo scattering, emissive volumes, and mixed scenes with complex lighting.

The plots show our method consistently produces less error than the baseline. Note that allowing more scattering events slows convergence in both methods. Scenes mixing volumes and surfaces under complex lighting are also more challenging. The general observation is that while producing lower error than the baseline, our longer term convergence speed slows down for multiple scattering and complex surface scenes. Interestingly, our long term convergence still shows clear advantage over the baseline for emissive volumes with multiple scattering. Overall, our method is superior in 1–10s time range, suggesting our approach may be applicable to previsualization for offline rendering.

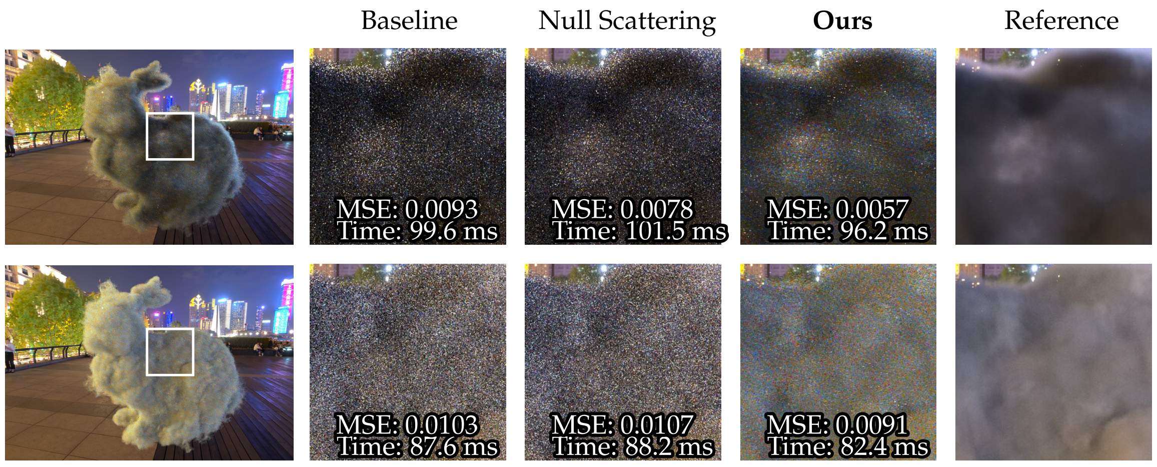


Figure 5.21: Comparing our baseline (decomposition tracking[5] plus residual ratio tracking[6]), a null scattering integration [9], and our method on a (top) highly anisotropic *Bunny Cloud* (Henyey-Greenstein scattering coefficient $g = 0.8$). (Bottom) For comparison, we show an isotropic *Bunny Cloud* under identical lighting. Images shown with 3-bounce multiple scattering.

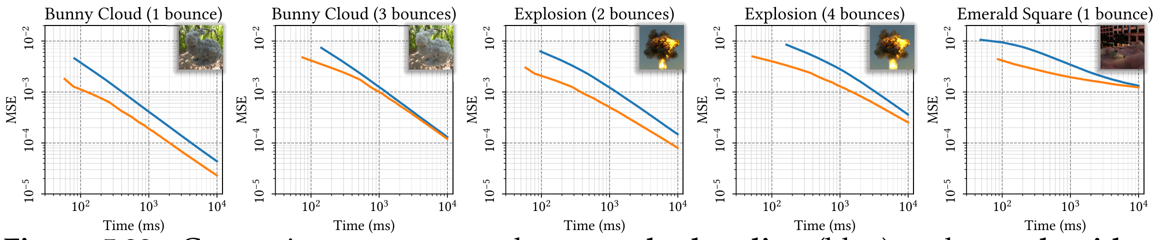


Figure 5.22: Comparing convergence between **the baseline** (blue) and **our algorithm** (orange) using log-log plots showing MSE vs. render time from 0 to 10 seconds.

5.6.5 Comparison with Vertex Reuse

Section 5.4.2 notes we can either resample paths by reusing path vertices $\mathbf{x}_i$ or by reusing directions $\boldsymbol{\omega}_i$. We chose to reuse directions; while costs are somewhat higher for direction reuse, it reduces noise fairly significantly. This is because paths act a little like virtual point lights under reuse, which leads to more singularities and fireflies. However, if these could be reduced, it may make sense to switch back to vertex reuse for the improved performance. See Figure 5.23 to compare visually between vertex and directional reuse.

5.6.6 Denoised Results

In Figure 5.24, we compare denoising applied to both our baseline and our resampling technique (using both single scattering and 7-bounce multiple scattering); both using the new OptiX [14] 7.3 temporal denoising mode, though recent work [154] may provide even

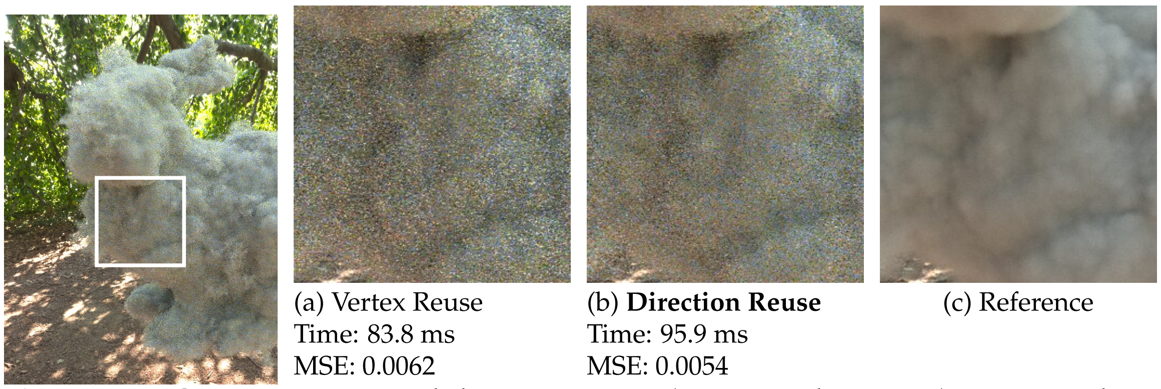


Figure 5.23: Compare vertex and direction reuse. (We reuse directions.) Here we show 7-bounce multiple scattering in the *Bunny Cloud* scene.

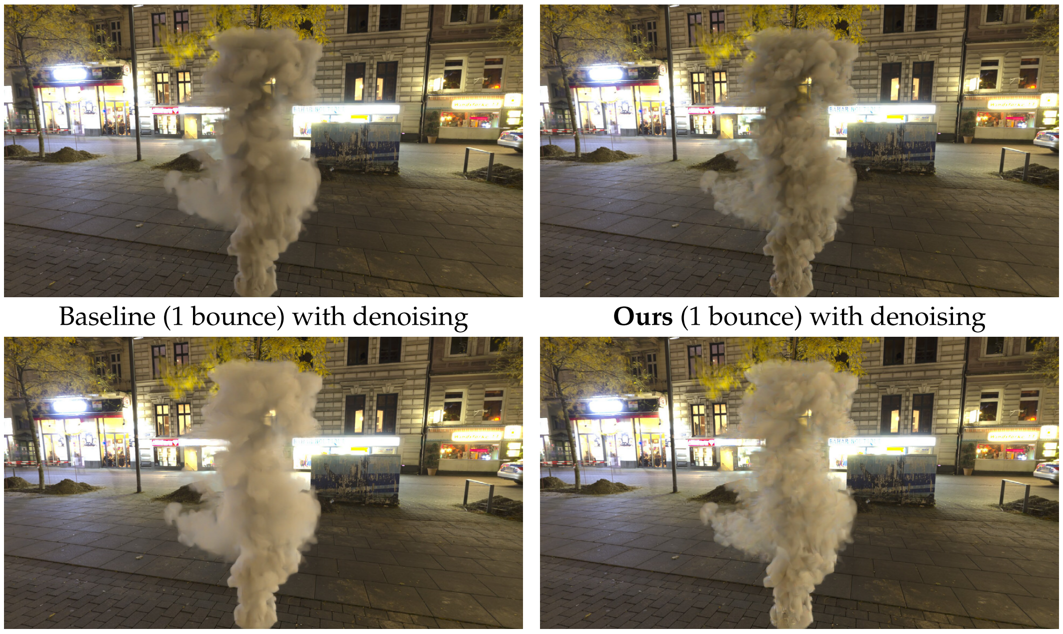

Baseline (1 bounce) with denoising
Ours (1 bounce) with denoising
Baseline (7 bounces) with denoising
Ours (7 bounces) with denoising

Figure 5.24: The OptiX 7.3 denoiser (with temporal denoising) applied to both our baseline and our method.

better denoising quality. While OptiX produces amazingly denoised results in both cases, the better sampling provided by our technique preserves much higher frequency details in the animation, while the baseline gives a smoother, more washed out look. Part of this is also due to the improved motion vectors we provide with our novel temporal reprojection plus velocity resampling. The supplementary video of our SIGGRAPH Asia 2021 paper compares the denoised results under animation.

5.7 Conclusion

We introduced a sampling solution extending resampled importance sampling [8] and ReSTIR [7] to path space, enabling real-time rendering of heterogeneous volumes in complex lighting environments. Resampling exposes new user-defined target PDFs in each reuse step. By adjusting these target PDFs, even with biased or approximate distributions, we can dramatically improve the distribution quality used to select our final pixel samples.

Beyond the prior resampling work, our approach extends resampling to multi-bounce paths on surfaces and in volumes, mixes path samples of varying lengths, and shows how transmittance estimates of increasing fidelity can be injected over multiple resampling steps. We jointly sample multiple dimensions during resampling; free-flight distances and scattering directions are mixed together, unlike prior work [7] that exclusively considers directions. We demonstrate an efficient GPU implementation that outperforms state-of-the-art.

Our work inherits some limitations of prior resampling [7] techniques. For instance, we exploit coherence between samples and perform poorly where no coherence exists. Specifically, high frequency variations (e.g., of lighting, density, motion) limit coherence across boundaries, increasing nearby variance.

Additionally, our work uses scalar target functions. This samples chroma channels identically, leaving color noise (e.g., Figure 5.19). Such noise is usually minor, except in scenes with different, highly-saturated lights. Using separate target functions per channel avoids this issue, but at substantial cost. Exploring efficient sampling to reduce color noise is interesting future work.

For sampling emission, a line integral which effectively combines our method with the FNEE method [93] may more efficiently collect radiance.

Another issue is the relative high cost for initial candidates and target function evaluation inside media, compared to Bitterli et al. [7]. Coarse volumes reduce cost at the expense of quality, less accurately approximating target functions. Future work may explore adaptive representations or ray marching to speed computations while minimizing quality loss. Such improvements will accelerate our work, enabling fast, many-bounce global lighting in the presence of complex lighting, volumes, and surfaces.

As in most modern real-time renderers, our volume rendering system provides input to a denoiser. But spatiotemporal reuse can introduce correlations in the noise, a characteristic not handled well by existing denoisers (e.g., in OptiX [14]). Finding additional ways to decorrelate the noise or better adapt the denoiser are interesting future directions.

CHAPTER 6

GENERALIZED RESAMPLED IMPORTANCE SAMPLING[1]

As scenes become ever more complex and real-time applications embrace ray tracing, path sampling algorithms that maximize quality at low sample counts become vital. Recent *resampling* algorithms building on resampled importance sampling (RIS) [17] reuse paths spatiotemporally to render surprisingly complex light transport with a few samples per pixel. These reservoir-based spatiotemporal importance resamplers (ReSTIR) and their underlying RIS theory make various assumptions, including sample independence. But sample reuse *introduces correlation*, so ReSTIR-style iterative reuse loses most convergence guarantees that RIS theoretically provides.

In this chapter, we introduce generalized resampled importance sampling (GRIS) to extend the theory, allowing RIS on correlated samples, with unknown PDFs and taken from varied domains. This solidifies the theoretical foundation, allowing us to derive variance bounds and convergence conditions in ReSTIR-based samplers. It also guides practical algorithm design and enables advanced path reuse between pixels via complex shift mappings.

We prototype a path-traced resampler (ReSTIR PT) that runs interactively on complex scenes, capturing many-bounce diffuse and specular lighting while shading just one path per pixel (Figure 6.1). With our new theoretical foundation, we can also modify the algorithm to guarantee convergence for offline renderers.

[1]D. Lin, M. Kettunen, B. Bitterli, J. Pantaleoni, C. Yuksel, and C. Wyman, "Generalized resampled importance sampling: Foundations of ReSTIR," ACM Transactions on Graphics (TOG), vol. 41, no. 4, to appear, 2022.

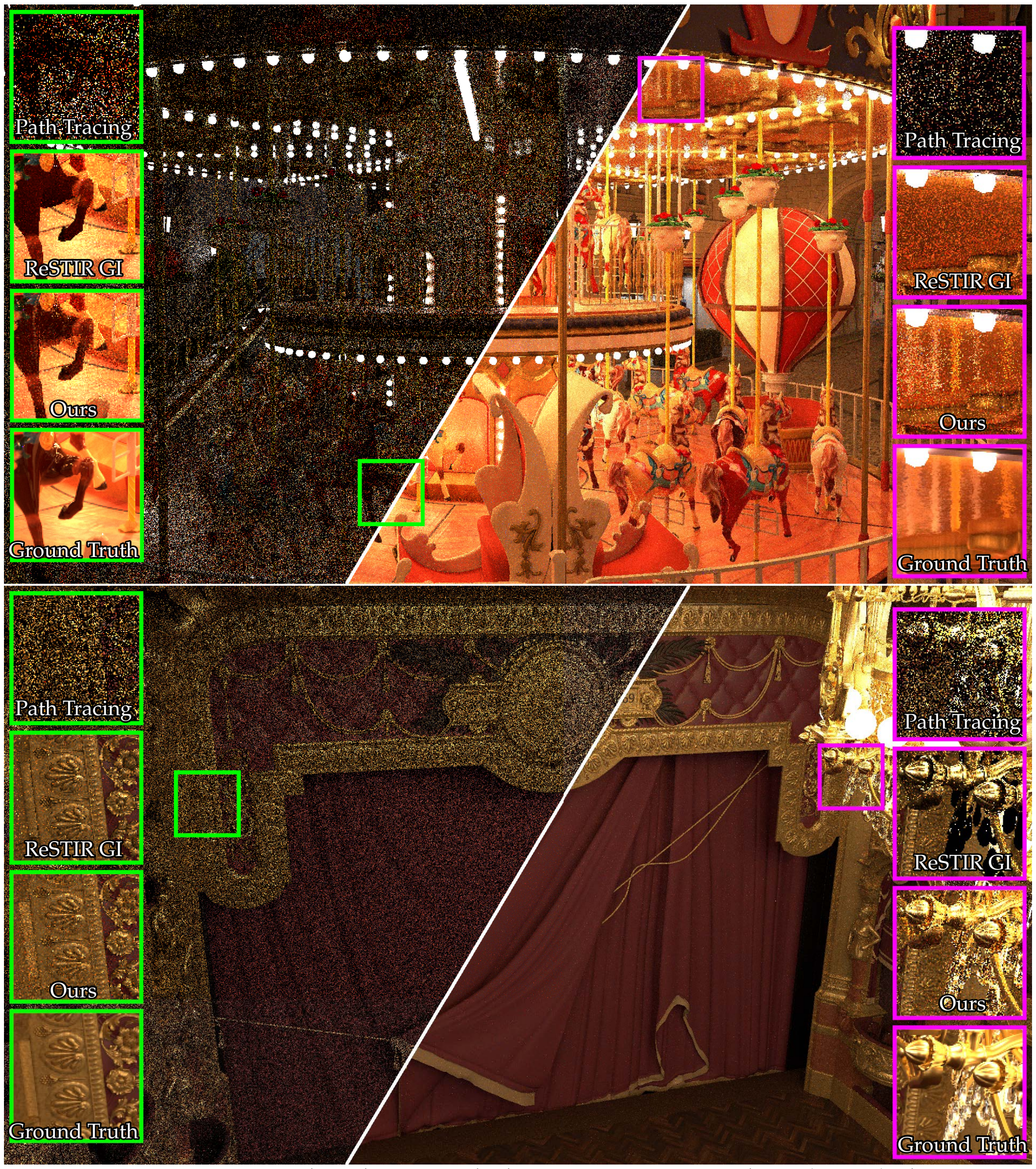


Figure 6.1: Our new generalized resampled importance sampling (GRIS) theory extends resampled importance sampling [8] to guarantee convergence even when applied to correlated samples arising from spatiotemporal reuse (i.e., Bitterli et al. [7]). GRIS allows applying ReSTIR to reuse arbitrary paths, shown with paths of length 10 in the *Carousel* (top) and *Paris Opera House* (bottom). Main images compare naive path tracing and our new ReSTIR PT in equal time (80 ms at 1920 $\times$ 1080). Insets show equal-time path tracing, ReSTIR GI [10], our ReSTIR PT, plus a converged reference. We significantly improve quality for glossy interreflection, reflections, refractions, and other high-frequency lighting. For *Carousel*, MAPE errors: path tracing (1.63), ReSTIR GI (0.45), and ReSTIR PT (0.39). Corresponding errors in *Opera House*: 1.28, 0.39, and 0.33. (*Carousel* ©carousel_world; *Paris Opera House* courtesy ©GoldSmooth from TurboSquid.)

6.1 Introduction

Monte Carlo algorithms form the core of modern rendering. While originally only feasible in offline renderers, ray-tracing hardware [13] has made such algorithms practical in real-time systems as well. However, strict real-time constraints in games limit feasible per-frame ray counts [155], giving many modern real-time path tracers budgets of *at most* one path per pixel.

Importance sampling reduces variance at low sample counts by improving sample distributions. But this becomes challenging for complex global lighting, e.g., Figure 6.1, where sampling from optimal distributions is impossible. Path guiding aims at on-line learning of complex distributions, but requires updating complex data structures [110] or neural models [156].

A new family of algorithms based on resampled importance sampling (RIS) [8] instead continually evolves a population of samples towards their optimal distribution via sample reuse within and across frames. Ideally, each sample converges to its "perfect" importance distribution given sufficient reuse. Such reservoir-based spatiotemporal importance resampling (ReSTIR) algorithms work for direct lighting [7], global illumination [10], and volume scattering [157]. ReSTIR leverages GPU parallelism via a streaming algorithm, reducing error up to $100\times$ compared to equal-time renderings without reuse.

However, convergence of these randomized distributions is poorly studied. Nabata et al. [158] approximate convergence for Talbot RIS with an upper bound, but only without sample reuse between pixels. Bitterli et al. [7] show these distributions are *unbiased*, but do not prove they converge in all circumstances.

In fact, in Figure 6.2 we show a trivial example where sample reuse, despite being unbiased, converges to a wrong result.

Ultimately, ReSTIR ignores a key issue: RIS assumes *independent and identically distributed* (i.i.d.) samples, often from a single source distribution. Reuse violates this independence, and ignoring the assumption slows convergence or causes divergence. Prior work empirically suggests sufficiently small correlation does not impede convergence [7, 10, 157, 159]. But it remains unclear if and when their correlation minimization efforts (e.g., randomizing reused spatial neighbors) guarantee convergence. When resampling for more complex lighting, maintaining sufficient decorrelation may be impossible without a

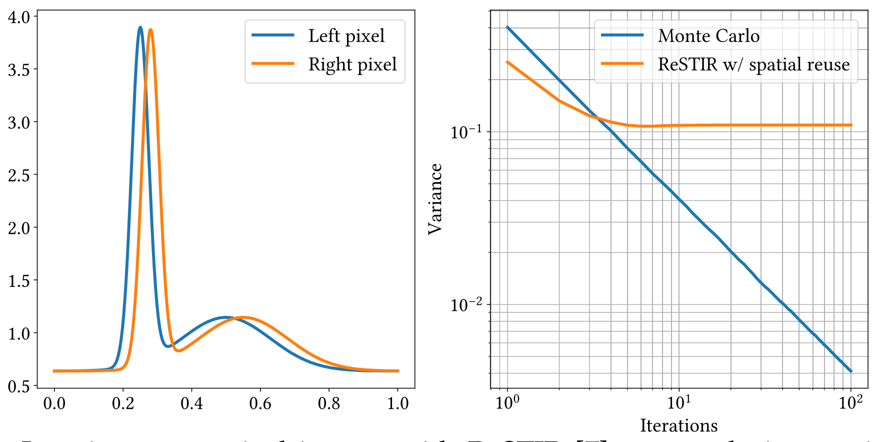


Figure 6.2: Imagine a two-pixel image, with ReSTIR [7] separately integrating two 1D functions (left). ReSTIR promises exponential growth in "effective" sample count at linear cost, but each ReSTIR iteration only adds two new independent samples; the rest of the exponential growth are duplicates. This makes the algorithm converge to the wrong result (right). Our GRIS theory explains when such cases occur and how to guarantee proper convergence.

deeper theoretical understanding.

We introduce *generalized resampled importance sampling* (GRIS), a new theoretical framework that lifts the i.i.d. assumption and helps understand, design, and discuss convergence for complex samplers, like ReSTIR. With GRIS, we can apply resampling to combine correlated candidate samples, drawn from potentially different domains and mapped to estimate a single integral (see Section 6.3).

Many derivations in Talbot [8] and Bitterli et al. [7] are special cases of our theory; we generalize prior work while proving conditions under which ReSTIR is unbiased and consistent.

Our main contributions include that we:

- Derive RIS with paths from other pixels by shift mappings,
- Give conditions for unbiasedness and convergence,
- Derive MIS weights satisfying the convergence constraints and help minimize variance, (Section 6.3.4),
- Explain how some prior ReSTIR design decisions, e.g., M-capping, are *vital* for ensuring convergence (Section 6.5.4),

- Show proper shift mappings help control noise when spatiotemporally reusing paths (Section 6.6),
- Design shift mappings with improved performance and quality by BSDF lobe specific connections (Sections 6.6.5 and 6.6.6),
- Apply GRIS theory to derive our ReSTIR PT that can reuse paths e.g., through glass.

Specifically, to guarantee convergence (see Section 6.4) when integrating a function f, one must:

- Use correct MIS weights during sample reuse,
- Select the target function $\hat{p}$ so $f/\hat{p}$ is not arbitrarily large,
- Control samples' resampling weights w_i so $\mathrm{Var}\left[\sum w_i\right] \to 0$,
- Ensure sufficient sample count across f's domain, specifically to have enough "canonical" samples (see Section 6.4.5), and
- When temporally resampling, use a reasonable M-cap to limit correlations between frames.

With our new theory and shift maps, we more efficiently reuse samples, obtaining a robust, unbiased light transport algorithm that can handle even very complex lighting scenarios while remaining fully amenable to efficient GPU parallelization and real-time use (see Figure 6.1). We also show that, without temporal resampling, ReSTIR can further be used to largely accelerate offline renderers.

While many proofs and derivations reside in Appendix B, Sections 6.3 and 6.4 remain mathematically dense. We have unlined key results throughout, and starred sections ($\star$) skippable by readers less interested in theory. For engineers, we suggest reading through Section 6.3.1 and then skipping to Section 6.6.

6.1.1 Chapter Roadmap

Table 6.1 shows a summary of notation used in this chapter.

In Section 6.2, we briefly review the state-of-the-art in resampled importance sampling theory and motivate the need for extending it.

Table 6.1: Summary of notation in the chapter

x, y	A general input to a function
$\bar{\mathbf{x}}$, $\mathbf{x}_i$	A path and a vertex i on the path
Ω_i	Domain from which samples are drawn
Ω	Domain of integration of our function f
X_i	Input sample for RIS, often a sequence $(X_i)_{i=1}^{M}$
Y	Sample Y selected via RIS
	($Y = X_s$ in the simple case or $Y = T_s(X_s)$ in general)
M, N	Number of input and output samples for RIS
$p_X(\cdot)$	Probability density of random variable X at a location
$p(\cdot)$	Shorthand for the above when the random variable is clear
$\hat{p}(\cdot)$	Unnormalized *target* distribution (we *aim* to select $Y \propto \hat{p}$)
$\bar{p}(\cdot)$	Normalized target PDF (i.e., $\bar{p} = \hat{p} / \|\hat{p}\|_1$)
$f(\cdot)$	Function to integrate (e.g., the path contribution function)
$g_i(\cdot)$	A contribution function for $X_i \in \Omega_i$ to integrate f in Ω
W_i	*Unbiased contribution weights*; estimate reciprocal PDFs
w_i	*Resampling weights*; RIS selects one X_i based on $w_i / \sum w_j$
c_i	*Contribution MIS weights*; prior works' MIS weights
m_i	Our new *resampling MIS weights*
$T_i(\cdot)$	A shift mapping; maps samples from domain Ω_i to Ω
$\left\| \frac{\partial T_i}{\partial x} \right\|$	Jacobian of shift mapping T_i
$\hat{p}_{\leftarrow i}(\cdot)$	"$\hat{p}$ from i." Generalizes $\hat{p}$ to include shift maps from Ω_i
C	Various constants, as bounds in convergence proofs
R, $\|R\|$	Canonical samples and their number

In Section 6.3, we present our new generalization of RIS to resample from multiple input domains Ω_i into a target domain Ω, using shift maps $T_i : \Omega_i \to \Omega$ analogous to those in gradient-domain rendering. We establish conditions under which GRIS unbiasedly integrates any function f defined over Ω, and conditions ensuring output sample distributions converge to the specified target resampling PDF $\bar{p}$.

In Section 6.4, we show integration error directly relates to the variance of a RIS normalization factor. When this variance disappears, GRIS becomes a zero-variance integrator. We can achieve this by taking additional samples from the current, *canonical* pixel (not just from distant neighbors), and using robust resampling MIS weights.

By configuring ReSTIR to obey the GRIS convergence constraints, in Section 6.5, we observe it becomes a non-Markovian chain, forever exploring path space with one sample per pixel. In a still scene, averaging frames converges, and real-time usage gives a single state of the chain each frame. Cross-frame correlations hinder convergence for offline rendering, but spatial reuse remains beneficial.

Section 6.6 designs shift mappings for cross-pixel path reuse, and presents several new shift modifications to improve efficiency. Section 6.7 discusses our implementation of ReSTIR PT, and Section 6.8 presents results and experimental validation.

Appendix B contains mathematical proofs and derivations, additional analysis, and more details about our implementation.

6.2 Resampled Importance Sampling Review

Before introducing GRIS in Section 6.3, we first review resampled importance sampling (RIS) using the notation and terminology of our generalized theory. Figure 6.3 highlights differences between existing theory, e.g., Talbot [8], and our new generalization.

The RIS Algorithm	**Talbot et al. [17]** *Identically distributed samples*	**Talbot [8]** *Differently distributed samples*	**GRIS [Ours]** *Correlations & different source domains*
1. Generate M initial candidate samples: $(X_1, \ldots, X_M)$	Samples from same domain: $X_i \in \Omega$ with same PDF p	Samples from same domain: $X_i \in \Omega$ with different PDF p_i	Samples from arbitrary domains: $X_i \in \Omega_i$; intractable p_i are OK
2. Evaluate their *unbiased contribution weights*: W_i	$W_i = 1/p(X_i)$	$W_i = 1/p_i(X_i)$	W_i must unbiasedly estimate $1/p_i(X_i)$
3. Evaluate their *resampling weights*: w_i	$w_i = \frac{1}{M}\hat{p}(X_i)W_i$	$w_i = m_i(X_i)\hat{p}(X_i)W_i$	$w_i = m_i(T_i(X_i))\,\hat{p}(T_i(X_i)) \cdot W_i\,\lvert\partial T_i/\partial X_i\rvert$
4. Select s proportionally to w_i and output Y in Ω	Simply output: $Y = X_s$	Simply output: $Y = X_s$	Output sample mapped from Ω_i to Ω: $Y = T_s(X_s)$

Figure 6.3: We generalize Talbot's [8] resampled importance sampling in various ways. (Red) Basic RIS assumes i.i.d. samples X_i, all drawn with one PDF p from the domain Ω of integrand f. (Blue) More advanced forms allow candidates with different PDFs $p_i(X)$, adding MIS terms m_i to remain unbiased. Sample reuse, as in ReSTIR [7], adds correlations between candidate samples X_i and requires using unbiased estimates of $1/p_i(X_i)$ for W_i. But current theory fails to guarantee convergence in these cases (e.g., Figure 6.2). (Green) Our new theory corrects this, providing convergence guarantees even with correlated candidate samples X_i from arbitrary domains Ω_i and differing, intractable PDFs p_i. The unbiased estimate for the integral of f is $f(Y)W_Y$ in all cases, with W_Y defined in Equation 6.2.

6.2.1 Identically Distributed Samples

Basic RIS takes as input a sequence of independent and identically distributed (i.i.d) random samples $(X_i)_{i=1}^{M}$ in some domain Ω, distributed with known PDF p. The goal is to randomly pick Y from the sequence so that its PDF, p_Y, constitutes a better importance sampler for integrating function f over Ω.

More precisely, we define a non-negative target function $\hat{p}$ and choose Y randomly such that, as the input sample count M grows, the realized PDF p_Y better and better approximates a normalized $\hat{p}$ (i.e., p_Y approximates $\bar{p}(x) = \hat{p}(x)/\|\hat{p}\|_1$).

Algorithmically, from inputs X_i we select one, $Y = X_s$, with probability $\Pr[s = i] = w_i / \sum_{j=1}^{M} w_j$, using resampling weights w_i. Prior work defines w_i as $\hat{p}(X_i)/p(X_i)$ (e.g., Bitterli et al. [7], Eq. 5). As weights are relative, selection probability is invariant to multiplicative constants, and we define

$$w_i = \frac{1}{M}\hat{p}(X_i)W_i \qquad \text{and} \qquad W_i = \frac{1}{p(X_i)} \tag{6.1}$$

for notational consistency. The PDF of the selected sample Y is intractable, but its *unbiased contribution weight*

$$W_Y = \frac{1}{\hat{p}(Y)} \sum_{i=1}^{M} w_i \tag{6.2}$$

can be used in place of $1/p_Y(Y)$ (e.g., Bitterli et al. [7], Eq. 12). Assuming $p_Y > 0$ where $f > 0$, i.e., $\operatorname{supp} f \subset \operatorname{supp} Y$, we have

$$\int_{\Omega} f(x)\,\mathrm{d}y = \mathbb{E}\left[f(Y)W_Y\right] \ . \tag{6.3}$$

Given appropriate constraints, p_Y converges to $\bar{p}$ and the variance $\operatorname{Var}[f(Y)W_Y]$ asymptotically approaches the variance expected if Y had PDF *exactly* $\bar{p}$. Choosing $\hat{p}$ proportional to f guarantees the estimate $f(Y)W_Y$ is, asymptotically, zero-variance.

6.2.2 Differently Distributed Samples

If the samples X_i have different PDFs p_i, the situation becomes more complex. This requires what we call *resampling MIS*, a partition of unity with weights m_i, for $m_i \geq 0$ and

$$\sum_{i=1}^{M} m_i(x) = 1 \tag{6.4}$$

for all x in $\hat{p}$'s support. Talbot [8] proposes weights analogous to Veach's [160] balance heuristic,

$$m_i(x) = \frac{p_i(x)}{\sum_{j=1}^{M} p_j(x)}. \tag{6.5}$$

The key algorithmic change is then replacing the $1/M$ term in w_i (Equation 6.1) with these MIS weights (see Figure 6.3, blue column), i.e.,

$$w_i = m_i(X_i)\, \hat{p}(X_i)\, W_i \qquad \text{and} \qquad W_i = \frac{1}{p_i(X_i)}\,. \tag{6.6}$$

Assuming at least one PDF p_i covers each $x \in \operatorname{supp} \hat{p}$, Equation 6.3 holds with W_Y from Equation 6.2 using these updated w_i. Convergence requires more assumptions than in Section 6.2.1, but is achievable (e.g., Section 6.4.7).

6.2.3 Why Generalize Resampling?

Early applications of RIS, e.g., for BSDF importance sampling, aim to choose a $\hat{p}$ that cheaply approximates f so that resampling from multiple cheaply generated candidates speeds convergence.

ReSTIR, however, *reuses* samples across pixels to amortize costs for simultaneous estimation of multiple integrals. With this goal, $\hat{p}$ need not be simpler than f, if reusing prior samples is cheaper than generating a new one. ReSTIR also gains efficiency if a reused sample's PDF better approximates the target integrand. Due to iterative use of RIS, in such cases, using $\hat{p} = f$ may be reasonable, especially for complex paths (e.g., Lin et al. [157]).

Talbot's RIS theory assumes independent samples X_i lying in a shared domain Ω. ReSTIR stretches these assumptions, so it may not retain *any* theoretical convergence guarantees. In fact, with seemingly innocuous algorithmic modifications, correlated reuse can cause convergence to a wrong result.

6.3 Generalized RIS

Our generalized resampled importance sampling (GRIS) allows mapping samples between domains and identifies the constraints for which this is unbiased and converges.

Unlike traditional RIS, which selects from independent samples in one domain, we allow potentially correlated inputs $(X_i)_{i=1}^{M}$ from different domains Ω_i. Generalized RIS randomly selects sample X_s and maps it to f's domain Ω via a *shift mapping*, $Y = T_s(X_s)$, so that the PDF of Y approaches target $\bar{p}$ (i.e., a normalized $\hat{p}$).

6.3.1 Overview

Before delving into theoretical details for our generalization, we briefly overview our approach and relate it to traditional RIS.

We assume input samples X_i, perhaps from varying domains Ω_i, need not be independent and are paired with unbiased contribution weights $W_i \in \mathbb{R}$ that can replace $1/p_i(X_i)$ for integration. This explicitly allows prior resampled inputs; while a resampled input X_i has an intractable PDF p_i, its weight W_i *is* tractable (i.e., Equation 6.2). We formalize unbiased contribution weights in Section 6.3.2.

To reuse samples to integrate f over Ω, we must map our random samples $X_s \in \Omega_s$ into Ω with a shift mapping $T_s : \Omega_s \to \Omega$. This shift modifies the PDF via the PDF transformation laws,[2] requiring the shift map's Jacobian determinant, $|\partial T_i/\partial x|$. We formalize shift mappings in Section 6.3.3.

Algorithmically, this changes various aspects of RIS (see Figure 6.3, green column). We must transform samples to a common domain Ω, so resampling weights include shift maps T_i and their determinants:

$$w_i = m_i(T_i(X_i))\, \hat{p}(T_i(X_i))\, W_i \cdot |\partial T_i/\partial X_i| \ . \tag{6.7}$$

We do not require tractable p_i; we may use $W_i = 1/p_i(X_i)$, but we may also use numerical contribution weights W_i from e.g., a prior RIS pass (Equation 6.2). Before using the selected sample for integration (or further resampling), we must shift it to the appropriate domain, i.e., our output sample is $Y = T_s(X_s)$.

Unbiased contribution weights W_Y for output Y are again given by Equation 6.2. With the constraints we derive below, p_Y converges to $\bar{p}$ such that $\mathrm{Var}[f(Y)W_Y]$ is guaranteed to approach $\mathrm{Var}[f(Y)/\bar{p}(Y)]$. This achieves asymptotic zero-variance integration with a single Y if $\hat{p} \propto f$.

6.3.2 Unbiased Integration with Generalized RIS

Again, we assume potentially correlated input samples $(X_i \in \Omega_i)_{i=1}^{M}$ with arbitrary source domains Ω_i. Furthermore, samples X_i must be paired with unbiased contribution weights W_i, acting as replacements for potentially intractable reciprocal PDFs $1/p_i(X_i)$.

[2]With $y = T(x)$, we have $P_Y(y) = \left|\frac{\partial P}{\partial y}\right| = \left|\frac{\partial P}{\partial x}\right| \left|\frac{\partial x}{\partial y}\right| = P_X(x) \left|\frac{\partial T}{\partial x}\right|^{-1}$.

We first derive an unbiased integrator for function f over domain Ω, assuming almost arbitrary resampling weights w_i. In Section 6.3.4, we replace these arbitrary w_i with weights that lead to asymptotic convergence to the target PDF $\bar{p}$.

We formally define unbiased contribution weights W_i as follows:

Definition 6.3.1. An *unbiased contribution weight* $W \in \mathbb{R}$ for a random variable $X \in \Omega$ is any real-valued random variable W for which

$$\mathbb{E}\left[f(X)W\right] = \int_{\text{supp}(X)} f(x)\,\mathrm{d}x \tag{6.8}$$

for any integrable function $f : \Omega \to \mathbb{R}$.

The expression $f(X)W$ generalizes the ratio $f(X)/p(X)$ in Monte Carlo integration: if p is tractable, we can use $W = 1/p(X)$. If not, as when picking X with RIS, these weights still allow unbiased integration. The integral is naturally limited to where $p > 0$, i.e., $\text{supp}(X)$. Similar definitions have been used outside computer graphics, e.g., Liang and Cheon [161].

Unbiased contribution weights naturally replace the reciprocal of the marginal PDF; in fact, they unbiasedly estimate it,

$$\mathbb{E}\left[W \mid X\right] = \frac{1}{p_X(X)}. \tag{6.9}$$

This is not coincidence, but equivalence. Any unbiased estimator for the inverse marginal PDF (Equation 6.9) is an unbiased contribution weight (Equation 6.8) and vice versa (Theorem A.1).

In RIS, we resample X_i proportionally to w_i. We need to express the contribution of chosen sample X_s that gives an unbiased estimate for the integral of f. To do this, we start by assigning each sample X_i a corresponding *contribution function* $g_i : \Omega_i \to \mathbb{R}$ that gets evaluated if selecting index $s = i$.

We then look at the expectation of $g_s(X_s)W_s$ divided by the RIS selection probability of index s. The PMF of the selection index is $p_s(i) = w_i / \sum_{j=1}^{M} w_j$, and with some caution,[3] we get

[3]Technically, this equation requires that $w_i > 0$ whenever $g_i(X_i) \neq 0$, but we will later introduce a partition of unity that lifts this requirement.

$$\begin{aligned}\mathbb{E}\left[\frac{g_s(X_s)W_s}{p_s(s)}\right] &= \mathbb{E}\left[\sum_{i=1}^{M} g_i(X_i)\frac{\cancel{p_s(i)}}{\cancel{p_s(i)}}W_i\right] = \\ &= \sum_{i=1}^{M}\int_{\mathrm{supp}(X_i)} g_i(x_i)\,\mathrm{d}x_i, \end{aligned} \tag{6.10}$$

The first step expands the expectation as a sum over the possible cases, and the second step utilizes the definition of unbiased contribution weights to transform the sum of expectations into a sum of integrals. RIS naturally skips sampling areas where the random variables have zero PDF, limiting integration to the supports of X_i. Equipped with this result, we can now proceed to transform the remaining sum of integrals into the desired integral of f by carefully choosing unknowns g_i.

Choosing g_i so the right-hand-side becomes the integral of f yields an unbiased contribution for the selected sample $Y = X_s$. In the special case that X_i are all from the same domain Ω and support S, and all w_i are positive in S, we can recover basic RIS by choosing $g_i = \frac{1}{M}f$ for all i, giving:

$$\mathbb{E}\left[\frac{1}{M}f(Y)\frac{\sum_{j=1}^{M} w_j}{w_s}W_s\right] = \int_{\mathrm{supp}(Y)} f(x)\,\mathrm{d}x. \tag{6.11}$$

Comparing to Equation 6.8, we observe that in this restricted case the expectation is of form $\mathbb{E}[f(Y)W_Y]$ with

$$W_Y = \frac{1}{M}\frac{\sum_{j=1}^{M} w_j}{w_s}W_s, \tag{6.12}$$

making W_Y an unbiased contribution weight for Y, i.e., $\mathbb{E}[f(Y)W_Y]$ integrates any function f over the support of Y. In Section 6.3.3, we extend our result to samples X_i coming from multiple domains Ω_i.

• **Degenerate case.** If all w_i are 0, no sample is selected and the contribution is zero. Intuitively, one may think of returning a zero-contribution null-sample $Y_\emptyset$ outside the sampling and integration domains (i.e., $\hat{p}(Y_\emptyset) = f(Y_\emptyset) = 0$). The value of $W_{Y_\emptyset}$ is then irrelevant, and can be set to zero.

6.3.3 Shift Mapping

In GRIS, samples X_i may originate from arbitrary domains Ω_i. To integrate $f : \Omega \to \mathbb{R}$ with samples $X_i \in \Omega_i$, we must transform the right-hand side of Equation 6.10 into the integral of f. To do this, we choose g_i that map X_i from Ω_i to Ω and evaluate f at the result.

Since a map from Ω_i to Ω changes the variables of integration, it must be bijective. Mappings between complicated domains can be non-trivial to construct, so we settle for a bijection from a subset of Ω_i to its image in Ω. As in prior work (e.g., Manzi et al. [162]), we call such bijections *shift mappings*, T_i, and associate one with each domain Ω_i.

Definition 6.3.2. A *shift mapping* T_i from Ω_i to Ω is a bijective function from a subset $\mathcal{D}(T_i) \subset \Omega_i$ to its image $\mathcal{I}(T_i) \subset \Omega$.

Intuitively, we should choose contribution functions

$$g_i(x) = c_i(y_i)\, f(y_i) \left| \frac{\partial T_i}{\partial x} \right|, \tag{6.13}$$

where y_i is shorthand for $T_i(x)$, *contribution MIS* weights $c_i : \Omega \to \mathbb{R}$ are an arbitrary partition of unity $\sum_{i=1}^{M} c_i(y) = 1$ for $y \in \Omega$, and $\left|\frac{\partial T_i}{\partial x}\right|$ is the Jacobian determinant of $x \mapsto y_i$. In principle, this implements

$$\sum_{i=1}^{M} \int_{\Omega_i} g_i(x)\,\mathrm{d}x = \int_{\Omega} f(x)\,\mathrm{d}x, \tag{6.14}$$

but care is required in the details; e.g., Equation 6.13 is not defined for $x \notin \mathcal{D}(T_i)$. We fix this by defining $g_i(x) = 0$ for $x \notin \mathcal{D}(T_i)$ and updating the contribution MIS weights c_i to compensate.

We assume weights w_i are arbitrary non-negative random variables related to target function $\hat{p}$ as follows: $w_i > 0$ iff $X_i \in \mathcal{D}(T_i)$ and $\hat{p}(Y_i) > 0$. Essentially, $w_i > 0$ when $Y_i = T_i(X_i)$ exists and is in the support of $\hat{p}$, otherwise $w_i = 0$ to avoid choosing X_i. Later, we slightly relax this constraint.

Under these assumptions, each possible Y must be in $\operatorname{supp} \hat{p}$ and be samplable as $Y = T_i(X_i)$ by one or more X_i that has positive PDF (i.e., $X_i \in \operatorname{supp} X_i$), and vice versa. Mathematically,[4]

$$\operatorname{supp} Y = \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i). \tag{6.15}$$

This implies $\operatorname{supp} Y \subset \operatorname{supp} \hat{p}$. Later, we assume $\operatorname{supp} \hat{p} \subset \operatorname{supp} Y$, which also implies $\operatorname{supp} Y = \operatorname{supp} \hat{p}$.

[4]If $\operatorname{supp} X_i$ is larger than the domain of T_i, we set $T_i(\operatorname{supp} X_i) = T_i(\operatorname{supp} X_i \cap \mathcal{D}(T_i))$.

Carefully substituting the above g_i, with $g_i(x) = 0$ if $x \notin \mathcal{D}(T_i)$, into the left-hand side of Equation 6.10, this gives the equality

$$\mathbb{E}\left[c_s(Y)f(Y)\left|\frac{\partial T_s}{\partial X_s}\right|\frac{\sum_{j=1}^{M} w_j}{w_s}W_s\right] = \int_{\operatorname{supp}(Y)} f(x)\,\mathrm{d}x, \tag{6.16}$$

where W_s is the unbiased contribution weight of X_s.

The constraints that contribution MIS weights c_i must fulfill for this to hold are that for all $y \in \operatorname{supp} Y$,

$$\sum_{\substack{i=1 \\ y \in T_i(\operatorname{supp} X_i)}}^{M} c_i(y) = 1. \tag{6.17}$$

Interpret this as: every realizable y, possibly from multiple Ω_i, must be covered exactly once in total. The summation only accounts for domains Ω_i from which y can be realized as $y = T_i(x_i)$ with non-zero PDF. In principle, negative values of c_i work, but later we find that only $c_i \geq 0$ allow chaining multiple passes of GRIS.

Again, the expectation in Equation 6.16 is of the form $\mathbb{E}[f(Y)W_Y]$ for arbitrary integrable function f in Ω, and the right-hand side integrates f over the support of Y, per the definition of unbiased contribution weights in Equation 6.8. This means that with

$$W_Y = c_s(Y)\left(W_s\left|\frac{\partial T_s}{\partial X_s}\right|\right)\left[\frac{\sum_{j=1}^{M} w_j}{w_s}\right], \tag{6.18}$$

$f(Y)W_Y$ unbiasedly estimates the integral of f over the support of Y, and that W_Y is an unbiased estimate for $1/p_Y(Y)$.

This specifies when generalized RIS can integrate an arbitrary function f: when the supports of random variates X_i (mapped to Ω via T_i) together cover the support of f.

This is automatically fulfilled if we choose one sampling domain, say Ω_1, as f's domain, use the identity shift $T_1(x) = x$ on Ω_1, and generate X_1 with an importance sampler designed for the integrand f, i.e., so that $p(x_1) > 0$ whenever $f(x_1) > 0$; we will later call such samples *canonical*. Since p_{X_1} is known, we can use the unbiased contribution weight $W_1 = 1/p_{X_1}(X_1)$.

We later show earlier ReSTIR samplers can, a posteriori, be built on these observations, but first Section 6.3.4 covers how to realize convergence to target density $\bar{p}$ by setting w_i with Equation 6.19. The corresponding W_Y is then given by Equation 6.22.

• **Relaxing constraints ⋆.** The condition that $w_i > 0$ when $\hat{p}(Y_i) > 0$, for $Y_i = T_i(X_i)$, can be relaxed by also allowing $w_i = 0$ when $c_i(Y_i) = 0$ or $W_i = 0$, i.e., when the expectation does not change. The validity of Equation 6.17 must be explicitly guaranteed in $\operatorname{supp} \hat{p} \cap \bigcup_i T_i(\operatorname{supp} X_i)$ to make Equation 6.15 hold. We derive these constraints and the unbiasedness of the estimator in Section B.3.1. Using the w_i from the next section removes the need for these constraints.

6.3.4 Asymptotically Perfect Importance Sampling ⋆

Above, we generalized RIS to multiple domains for unbiased integration with near-arbitrary weights. Like Talbot's RIS, the goal of GRIS is producing samples following a desired distribution; we want the marginal probability density p_Y of output sample Y to converge to $\bar{p}$ as the input sample count approaches infinity.

We show this occurs with the following resampling weights:

$$w_i = \begin{cases} m_i(T_i(X_i))\, \hat{p}(T_i(X_i))\, W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right|, & \text{if } X_i \in \mathcal{D}(T_i) \\ 0, & \text{otherwise} \end{cases}, \tag{6.19}$$

given resampling MIS weights m_i and unbiased contribution weights W_i. As normalizing w_i gives resampling probabilities, w_i must be non-negative. It follows that m_i and W_i must also be non-negative (proofs in Section B.3.2), which we assume hereafter.

Weight w_i will be zero outside of $\operatorname{supp} \hat{p}$, as $\hat{p} = 0$. Requirements for m_i are similar to those for c_i. For all y in $\operatorname{supp} Y$,

$$\sum_{\substack{i=1 \\ y \in T_i(\operatorname{supp} X_i)}}^{M} m_i(y) = 1, \tag{6.20}$$

but we also require $m_i \geq 0$. The sum only includes indices that can generate y with a positive PDF. Unbiased integration also requires $m_i(y) > 0$ whenever $c_i(y) \neq 0$ so the m_i do not invalidate the partition of unity formed by c_i (proof in Section B.3.3).

Directly substituting w_i from Equation 6.19 into Equation 6.18 yields the unbiased contribution weight for new sample Y,

$$W_Y = \left[\frac{c_s(Y)}{m_s(Y)}\right] \frac{1}{\hat{p}(Y)} \sum_{j=1}^{M} w_j. \tag{6.21}$$

The condition that $m_i(y) > 0$ when $c_i(y) \neq 0$ now naturally avoids division by zero. Next, we show that choosing $m_i = c_i$ is ideal, naturally fulfilling this requirement.

W_Y in Equation 6.21 has multiple sources of variance. The sum $\sum_{j=1}^{M} w_j$ varies with inputs X_j and ratio $c_s(Y)/m_s(Y)$ varies with index s. Asymptotically approaching the desired sample density $\bar{p}$ requires the sum variance $\mathrm{Var}[\sum_{j=1}^{M} w_j]$ to approach zero. Even if fixing this sum as a constant, ratio $c_s(Y)/m_s(Y)$ can add significant variance. This disappears by selecting $c_i(y) = m_i(y)$.

Our improved and simpler unbiased contribution weight for GRIS becomes

$$W_Y = \frac{1}{\hat{p}(Y)} \sum_{j=1}^{M} w_j, \tag{6.22}$$

which we used to reformulate traditional RIS (Equation 6.2) to prepare for this generalization. We use this expression for W_Y hereafter.[5]

This allows deriving Theorem A.2 in Section B.1, which guarantees asymptotic convergence of p_Y, the PDF of resampled Y, to $\bar{p}$: consider the behavior of a sequence of resampling results Y_M (with $\operatorname{supp} \hat{p} \subset \operatorname{supp} Y_M$) as M increases. If variance of the summed resampling weights goes to zero,

$$\mathrm{Var}\left[\sum_{i=1}^{M} w_{M,i}\right] \xrightarrow{M\to\infty} 0, \tag{6.23}$$

then $\bar{p}(Y)/p_Y(Y)$ converges to 1 in the mean-square sense.[6] This sum approximates the integral of $\hat{p}$:

$$\mathbb{E}\left[\sum_{i=1}^{M} w_i\right] = \mathbb{E}\left[\hat{p}(Y) W_Y\right] = \int_{\operatorname{supp} Y} \hat{p}(y)\,\mathrm{d}y = \|\hat{p}\|. \tag{6.24}$$

Increasing input sample count M adds more terms to the sum; this tends to make each w_i smaller. Convergence of p_Y to $\bar{p}$ is subject to $\sum_{i=1}^{M} w_i$ approaching $\|\hat{p}\|$ as $\mathrm{Var}\left[\sum_{i=1}^{M} w_i\right] \to 0$, i.e.,

$$W_Y = \frac{1}{\hat{p}(Y)} \sum_{j=1}^{M} w_j \approx \frac{\|\hat{p}\|}{\hat{p}(Y)} = \frac{1}{\bar{p}(Y)} \quad \text{for large } M\,. \tag{6.25}$$

The guarantee from Equation 6.23 is quite strong. While convergence of p_Y to $\bar{p}$ may not be pointwise (new samples may introduce temporary fluctuations), the probability of

[5]The unbiased contribution weights in Equation 6.22 skip the division by M often seen in RIS and ReSTIR formulas, as our weights w_j already include this factor in the resampling MIS weights m_i (Equation 6.19); selecting $m_i = 1/M$ gives the prior formulations.

[6]Mathematically, this means $\mathbb{E}\left[\left|\frac{\bar{p}(Y_M)}{p_Y(Y_M)} - 1\right|^2\right] \xrightarrow{M\to\infty} 0$.

errors of any given size approaches zero, and each subset of Ω will, asymptotically, receive the correct ratio of samples.

In addition, as shown in Section 6.4.2, integration variance also goes to zero (if $\hat{p} \propto f$), i.e., in the limit we get the variance expected if Y were *exactly* distributed with target PDF $\bar{p}$.

6.4 Convergence and Variance Analysis $\star$

Above we presented a new GRIS theory and conditions for asymptotic convergence to a target distribution, but we have yet to discuss its asymptotic behavior as an integral estimator, particularly for Monte Carlo sampling. As infinite sample counts are impractical, we also want to analyze variance when using finitely many samples.

6.4.1 Reasonable Distributions $\star$

Before studying variance, we start by formally defining a *reasonable* importance sampling distribution:

Definition 6.4.1 (Reasonable distribution)**.** We say a PDF p is a reasonable importance sampling distribution for a non-negative function f (or p is reasonable for [integrating] f) if a bound C_f exists such that

$$f(x) \leq C_f \, p(x) \quad \text{for all } x. \tag{6.26}$$

We also say a random variate X with unbiased contribution weight W_X is reasonably distributed for f, if there exists a bound C_f

$$f(X)W_X \leq C_f \quad \text{with probability 1.} \tag{6.27}$$

Essentially, a reasonable distribution guarantees bounded Monte Carlo contributions. In standard Monte Carlo, $f(X)/p(X) \leq C_f$, and for importance sampling with unbiased contribution weights, $f(X)W_X \leq C_f$.

6.4.2 Asymptotic Variance of Integral Estimation $\star$

Asymptotic convergence of distribution p_Y naturally gets reflected in integration variance. Assuming $\bar{p}$ is a reasonable distribution for function f, then the unbiased integral estimate $f(Y)W_Y$ asymptotically has variance due only to any mismatch of $\bar{p}$ and f. If

$\hat{p}$ is chosen proportional to f, then $\bar{p} \propto f$ and estimate $f(Y)W_Y$ will be asymptotically zero-variance.

We formalize this in Theorem A.3. Convergence of p_Y to $\bar{p}$ is provided by Theorem A.2, and if $0 \le f \le C_f\,\hat{p}$ for some $C_f > 0$, then

$$\operatorname{Var}\left[f(Y)W_Y\right] \xrightarrow{M\to\infty} \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right], \tag{6.28}$$

where X has density $\bar{p}$. The zero-variance result follows naturally if $f(x)/\bar{p}(x)$ is constant.

A key takeaway is that if $\hat{p}$ is not proportional to f, increasing the input sample count eventually leads to diminishing returns; further variance reduction requires choosing a $\hat{p}$ better matching f.

6.4.3 Variance in the Finite Case ⋆

Above, we studied the asymptotic behavior of GRIS as sample count increases without bound. In practice, we are limited to finite M, so we aim to minimize variance in some computation budget. Fortunately, we may give explicit variance bounds for our integral estimate:

Theorem 1. *With the assumptions of Theorem A.3,*

$$\operatorname{Var}\left[f(Y)W_Y\right] \le \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] + b, \tag{6.29}$$

where X is distributed with density $\bar{p}$ and

$$b = C_f^2 \operatorname{Var}\left[\sum_{i=1}^{M} w_i\right]^{\frac{1}{2}} \left(\|\hat{p}\| + 2\operatorname{Var}\left[\sum_{i=1}^{M} w_i\right]^{\frac{1}{2}}\right). \tag{6.30}$$

Proof. Section 6.4.3. □

Here, C_f is the bounding constant for a reasonable distribution $\hat{p}$ for f. Theorem 1 says resampling converges to $\bar{p}$ by decreasing $\operatorname{Var}[\sum_{i=1}^{M} w_i]$, which acts as a concrete proxy for the current convergence state. As $\operatorname{Var}[\sum_{i=1}^{M} w_i]$ approaches zero, remaining variance stems from potential mismatches between $\hat{p}$ and f.

The law of total variance provides another decomposition of variance; applied to the expression $f(Y)W_Y$, we get

$$\mathrm{Var}[f(Y)W_Y] = \mathrm{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] + \mathbb{E}\left[f(Y)^2\,\mathrm{Var}\left[W_Y \mid Y\right]\right], \tag{6.31}$$

which says variance has two sources: mismatches between f and marginal density p_Y, and $\mathrm{Var}\left[W_Y \mid Y\right]$, the mean squared deviation of W_Y from its conditional expectation $1/\bar{p}(Y)$. Intuitively, if W_Y approaches $1/\bar{p}(Y)$ such that $\mathrm{Var}[W_Y \mid Y]$ goes to zero, $\mathrm{Var}[f(Y)W_Y]$ also approaches $\mathrm{Var}[f(Y)/\bar{p}(Y)]$.

For the special case of equality $f(x) = C_f\hat{p}(x)$, we can derive the exact variance

$$\mathrm{Var}\left[f(Y)W_Y\right] = \mathrm{Var}\left[\frac{f(Y)}{\hat{p}(Y)}\sum_{i=1}^{M} w_i\right] = C_f^2\,\mathrm{Var}\left[\sum_{i=1}^{M} w_i\right]. \tag{6.32}$$

Independent of $\hat{p}$'s proportionality to f, our analysis shows the importance of reducing $\mathrm{Var}[\sum_i w_i]$ given finite samples. In practice, we can minimize variance by making w_i more uniform. In particular, preventing singularities in w_i avoids unbounded variance.

6.4.4 Avoiding Singularities ⋆

Generalized RIS does not automatically remove singularities from Monte Carlo integration. As usual, avoiding large outliers requires additional guarantees. More concretely, if $\sum_{i=1}^{M} w_i$ is unbounded, the contribution

$$f(Y)W_Y = \frac{f(Y)}{\hat{p}(Y)}\sum_{i=1}^{M} w_i \tag{6.33}$$

can also be unbounded.

Clearly, very large w_i are detrimental to our goal of bringing $\mathrm{Var}[\sum_{i=1}^{M} w_i]$ to zero; we should aim to make the weight sum,

$$\sum_{i=1}^{M} w_i = \sum_{\substack{i=1 \\ X_i \in \mathcal{D}(T_i)}}^{M} m_i(T_i(X_i)) \cdot \hat{p}(T_i(X_i))W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right|, \tag{6.34}$$

as uniform as possible. Going through the terms, we identify potential challenges to maintaining uniformity:

1. Some samples y may be reachable via only finitely many T_i even in the limit

2. Resampling MIS weights m_i can greatly exceed $1/M$

3. The product $\hat{p}(T_i(X_i))W_i$ can be unbounded

4. Jacobians can be unbounded

These can be tackled one-by-one, e.g., (1) adding samples from all source domains when increasing M, and (4) modifying shift map domains to cut off extreme Jacobians (while maintaining bijectivity).

Instead, we simultaneously solve all four by designing a suitable sampling scheme and two robust MIS weight families. This guarantees bounded contributions, asymptotic convergence to $\hat{p}$, and realizes asymptotic zero-variance integration with samples from multiple domains.

6.4.5 Canonical Samples $\star$

Designing MIS weights for samples arising from multiple strategies normally requires knowing PDFs for samples with all strategies. Resampling gives up access to such PDFs. Instead, we design robust MIS weights by assuming samples X_i are associated with non-negative unnormalized target distributions $\hat{p}_i$, much like $\hat{p}$, that act as proxies for p_{X_i}.

RIS, and our generalization, conceptually need an infinite stream of random samples to asymptotically converge to desired distribution $\bar{p}$. If a subset of $\operatorname{supp}\hat{p}$ is covered by this stream only finitely many times, p_Y can not generally converge to $\bar{p}$ in this subset.

Sometimes, as in light transport, covering the support of $\hat{p}$ with samples X_i from *other* domains Ω_i may be challenging. By taking samples from an importance sampler that directly targets $\hat{p}$, we can cover $\operatorname{supp}\hat{p}$ as many times as needed for convergence. We define samples X_i that directly target $\hat{p}$ in Ω with the identity shift map and $\hat{p}_i = \hat{p}$ to be *canonical*, motivated by Bitterli [149], and present the following mathematical definition:

Definition 6.4.2 (Canonical Sample)**.** An input sample $X_i \in \Omega_i$ is *canonical* if its domain is Ω, it uses the identity shift map $T_i(x) = x$, uses $\hat{p}_i = \hat{p}$, and covers $\operatorname{supp}\hat{p}$ (i.e., $\operatorname{supp}\hat{p} \subset \operatorname{supp} X_i$).

We denote the set of indices of canonical samples in $1, \ldots, M$ by R and their number by $|R|$. Later, we find that if canonical sample count increases sufficiently as the total in-

put count increases, the MIS weights in Section 6.4.6 guarantee asymptotic convergence of p_Y to $\bar{p}$ when resampling from multiple domains.

6.4.6 Designing Robust MIS Weights $\star$

As motivated in Section 6.4.4, we design resampling MIS weights to guarantee bounded contribution for the chosen sample Y, assuming input samples X_i are reasonably distributed for target functions $\hat{p}_i$. To simplify the derivation, we define a new symbol, "$\hat{p}$ from i",

$$\hat{p}_{\leftarrow i}(y) = \begin{cases} \hat{p}_i\left(T_i^{-1}(y)\right)\left|T_i^{-1\prime}\right|(y), & \text{if } y \in T_i(\operatorname{supp} X_i) \\ 0 & \text{otherwise} \end{cases}, \tag{6.35}$$

i.e., for the sample y, evaluate its proxy PDF $\hat{p}_i$ at the sample location x in the original domain Ω_i, multiplied by the Jacobian determinant of the shift.

We aim to bound the resampling weights w_i and construct two families of MIS weights that guarantee this. We then derive their upper bounds, which decrease with additional canonical samples. Later, we utilize these bounds to guarantee convergence of p_Y to $\bar{p}$.

• **Generalized Talbot MIS.** The first family, which we derive in Section B.4, generalizes the weights of Talbot [8] into the following:

$$m_i(y) = \frac{\hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} \hat{p}_{\leftarrow j}(y)}. \tag{6.36}$$

Talbot's [8] form is obtained by assuming independent samples over one domain ($\Omega_i = \Omega$, $T_i(x) = x$) and using exact PDFs p_k in place of $\hat{p}_{\leftarrow k}$. This MIS family is analogous to the balance heuristic [160] between possible sources of sample Y.

• **Generalized pairwise MIS.** The second family of MIS weights, derived in Section B.4, generalizes Bitterli's [149] pairwise MIS, originally given for a single canonical sample and domain ($|R| = 1$, $\Omega_i = \Omega$, $T_i(x) = x$) for the *defensive* variant below. The key benefit of pairwise MIS is a significant cost reduction from $O(M^2)$ to $O(M\,|R|)$. This comes from restricting application of MIS to individual pairs of target functions, each involving only the target $\hat{p}$ and the source $\hat{p}_{\leftarrow i}$, if i is not a canonical sample, and an average of MIS between pairs $(\hat{p}, \hat{p}_{\leftarrow j})$ otherwise. We discuss the generalized pairwise MIS family more in Section B.4, but present here the *uniform* variant, which gives all inputs equal weight if they have the same $\hat{p}_{(\leftarrow)}$ values,

$$m_i(y) = \begin{cases} \frac{1}{M-|R|} \sum\limits_{j \notin R} \frac{\hat{p}(y)}{|R|\hat{p}(y)+(M-|R|)\hat{p}_{\leftarrow j}(y)}, & \text{if } i \in R \\ \frac{\hat{p}_{\leftarrow i}(y)}{|R|\hat{p}(y)+(M-|R|)\hat{p}_{\leftarrow i}(y)}, & \text{if } i \notin R \end{cases}, \tag{6.37}$$

and a slightly less efficient but often more robust *defensive* variant, which always gives canonical samples higher MIS weights than non-canonical samples,

$$m_i(y) = \begin{cases} \frac{1}{M} + \frac{1}{M} \sum\limits_{j \notin R} \frac{\hat{p}(y)}{|R|\hat{p}(y)+(M-|R|)\hat{p}_{\leftarrow j}(y)}, & \text{if } i \in R \\ \frac{M-|R|}{M} \frac{\hat{p}_{\leftarrow i}(y)}{|R|\hat{p}(y)+(M-|R|)\hat{p}_{\leftarrow i}(y)}, & \text{if } i \notin R \end{cases}. \tag{6.38}$$

• **Resampling weight bounds.** With these definitions, we can guarantee resampling weights w_i stay bounded (Theorem A.4): If the m_i are given by Equation 6.36, 6.37 or 6.38, and a sample X_i is reasonably distributed for integrating $\hat{p}_i$, i.e., $\hat{p}_i(X_i)W_i \leq C_i$ for some C_i, then the resampling weight of X_i is bounded as

$$w_i \leq \frac{C_i}{|R|}. \tag{6.39}$$

The condition that $\hat{p}_i(X_i)W_i$ is bounded is equivalent to a bounded relative error of W_i from its ideal value $1/\bar{p}_i(X_i)$. Starting from independent samples, we can guarantee this inductively:

• **Bounded variance.** If we independently sample X_i with a reasonable importance sampling strategy for target function $\hat{p}_i$ (i.e., $\hat{p}_i(X_i) \leq C_i\, p_i(X_i)$), then $\hat{p}_i(X_i)W_i = \hat{p}_i(X_i)/p_i(X_i) \leq C_i$, and Equation 6.39 applies to X_i.

If all input samples X_i fulfill $\hat{p}_i(X_i)W_i \leq C_i$ for constants C_i, then

$$\hat{p}(Y)W_Y = \sum_{i=1}^{M} w_i \leq \sum_{i=1}^{M} \frac{C_i}{|R|}, \tag{6.40}$$

and Equation 6.39 applies to Y, since $\hat{p}(Y)W_Y$ is bounded.

Inductively, chaining GRIS by starting from independent samples retains a finite worst-case resampling weight sum and a finite worst-case contribution $f(Y)W_Y$, assuming $f/\hat{p}$ is bounded.

A bounded random variable like $f(Y)W_Y$ features finite variance. Further, averaging such variables converges to the expectation. This important property is not automatically guaranteed by the resampling MIS weights earlier ReSTIR work implicitly used, as $m_i = 1/M$ often fails to properly partition unity, and additionally may not account for the singularities in Equation 6.33.

• **Constant resampling weights.** Constant resampling MIS weights $m_i(y) = 1/M$ are very cheap to evaluate. But such MIS weights only sometimes fulfill the constraints of m_i (Equation 6.20), i.e., when any realizable sample X_i could have been sampled from all other domains Ω_j with positive PDF.[7] This is generally not true, but an important exception is when all the input samples are canonical. This can be achieved e.g., by producing the input samples with GRIS that uses at least one canonical sample each time, and one of the resampling MIS schemes above. If constant weights are still used in the general case, convergence to $\bar{p}$ is lost along with our convergence and variance results. Bias can still be removed by using proper contribution MIS, as proposed by Bitterli et al. [7].

• **Tractable PDFs.** If all input samples have known, tractable PDFs (e.g., X_i come from importance samplers with known PDFs p_i), the generalized pairwise and generalized Talbot MIS weights can be modified to use p_i instead of $\hat{p}_i$, with instances of $\hat{p}$ replaced with PDF p_c of a fixed canonical sample X_c. The canonical samples must have a PDF reasonable for integrating $\hat{p}$. See Section B.4.4 for more information.

6.4.7 Guaranteeing Convergence ⋆

So far, we showed GRIS achieves asymptotic convergence of p_Y to $\bar{p}$ simply by requiring $\operatorname{Var}\left[\sum_{i=1}^{M} w_i\right] \to 0$. In this section we show how to guarantee convergence in a direct application of generalized RIS theory. Section 6.5 extends this analysis to multi-pass algorithms that guarantee convergence in a streaming manner, requiring only finite memory and amortizing computation between multiple integrals.

• **Independent samples.** We assume the case of multiple domains with robust resampling MIS weights (Section 6.4.6), applying Theorem A.4 to obtain a bound $w_i \le C_i/|R|$ on the resampling weights. If we also assume that pairs (X_i, W_i) are independent, then the w_i are independent, and $\operatorname{Var}\left[\sum_{i=1}^{M} w_i\right] = \sum_{i=1}^{M} \operatorname{Var}[w_i]$. We bound the variances by Popoviciu's inequality as

$$\sum_{i=1}^{M} \operatorname{Var}[w_i] \le \sum_{i=1}^{M} \frac{1}{4} \frac{C_i^2}{|R|^2}, \tag{6.41}$$

which converges to zero if $|R|$ grows fast enough compared to M and C_i. A practical constraint asserts the importance sampling quality of additional samples does not grow

[7]This alone is not generally enough to guarantee finite variance.

worse without bound, i.e., there exists an upper bound C such that $C_i \leq C$ for all i. Then we get $\text{Var}\left[\sum_{i=1}^{M} w_i\right] \leq C^2 \frac{M}{4|R|^2}$, guaranteeing convergence to zero if $|R|$ grows faster than $\sqrt{M}$ such that $|R|/\sqrt{M} \to \infty$. For example, $|R| \approx c\,M^{0.5001}$ for some $c > 0$ converges (slowly) in the limit. But more practically, we may ensure the ratio of canonical samples, $|R|/M$, never falls below some constant $\gamma > 0$ for large enough M; this guarantees a worst-case convergence rate of $O(1/M)$ in terms of variance.

• **Dependent samples.** Our GRIS theory does not assume sample independence; convergence and variance results only assume that $\text{Var}\left[\sum_{i=1}^{M} w_i\right] \to 0$ is true. For independent samples, this constraint is easy to prove. For dependent samples, this constraint may not be true. An easy counter-example uses duplicate samples X_i; no variance reduction can occur with increased sample count.

We still get convergence if sample correlation is weak enough. Assume the case of $|R|/M \geq \gamma$ and $w_i \leq C/|R|$ for all i (e.g., a single-domain with $m_i = 1/M$,[8] or multi-domains with our novel m_i). While this may not converge generally, we can guarantee convergence by assuming correlations between resampling weights w_i and w_{i+k} tend to zero as $k \to \infty$. More precisely, we assume there exists a non-negative sequence b_k such that regardless of i, the correlation $\rho_{i,i+k} \leq b_k$, and $b_k \to 0$. Then, we can manipulate

$$\text{Var}\left[\sum_{i=1}^{M} w_i\right] = \sum_{i=1}^{M} \text{Var}\left[w_i\right] + 2\sum_{i=1}^{M}\sum_{k=1}^{M-i} \text{Cov}(w_i, w_{i+k}), \tag{6.42}$$

where the first term converges to zero by the argument following Equation 6.41, and for the second term we derive in Section B.2

$$\sum_{i=1}^{M}\sum_{k=1}^{M-i} \text{Cov}(w_i, w_{i+k}) \leq \frac{C^2}{4\gamma^2}\left(\frac{1}{M}\sum_{k=1}^{M} b_k\right) \xrightarrow{M\to\infty} 0, \tag{6.43}$$

where γ is the minimum ratio of $|R|/M$ and C is an upper bound for all the C_i. The mean of b_k converges to zero since b_k converges to zero, and we get convergence with dependent samples.

Section B.2 also contains a generalized result allowing the ratio $|R|/M$ to decrease as M grows if the maximum correlation b_k falls quickly enough to compensate. For example, if $\sum_{k=1}^{\infty} b_k < \infty$, we can guarantee convergence with $|R| \geq c \cdot M^{0.5001}$ for some $c > 0$.

[8]If $\hat{p}(X_i)W_i \leq C$ for a single-domain with constant MIS, then $w_i = \hat{p}(X_i)W_i/M \leq C/M = C/|R|$ if all samples are canonical.

6.5 Amortization Over an Image with ReSTIR

Here, we reformulate two ReSTIR variants on top of generalized RIS: a novel progressive offline renderer and a reformulation of Bitterli et al. [7] using GRIS. We also reinterpret ReSTIR as an unbiased explorative non-Markovian chain and explain when averaging the images produced with it converges to the ground truth.

In light transport, we aim to produce an image where each pixel's color I_i is determined by integrals

$$I_i = \int_{\Omega} h_i(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}, \tag{6.44}$$

for pixel index i, paths Ω from sensor to a light, image filter h_i, and path contribution function f. A path $\bar{\mathbf{x}}$ generally contributes to few pixels due to filter h_i.

Common filtering methods sample paths for each pixel and *splat* the path contributions $f(X)/p_X(X)$ onto the image with kernel h_j. This allows, without loss of generality, integrating pixel i only over paths Ω_i *directly* contributing to it. This corresponds to using a box filter for h_i, but generalizing to more complex filters is straightforward.

Integrating I_i for each pixel only over its domain Ω_i, gives

$$I_i = \int_{\Omega_i} f(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}, \tag{6.45}$$

and sharing paths between domains Ω_i and Ω_j is impossible without path modification. We aim to more efficiently share paths between integrals by incorporating shift mappings into our resampling.

6.5.1 Formulation

We associate pixels i with path space domains Ω_i, integrands f_i (i.e., f restricted[9] to Ω_i), and target functions $\hat{p}_i$, which could be e.g., grayscale path contribution functions $|f_i|$ or still cheaper approximations with bounded relative error.

We assume that each pixel i is equipped with a sampler for canonical paths X_i that are reasonable for integrating $\hat{p}_i$. The samples could e.g., be directly importance sampled for $\hat{p}_i$, or resampled with RIS from multiple reasonably importance sampled initial candidates.

[9]Domain restriction: $\mathcal{D}(f_i) = \Omega_i \subset \Omega$ and $f_i(x) = f(x)$ in Ω_i.

6.5.2 Reservoirs and Weighted GRIS

We slightly extend the discussion on reservoirs in Section 5.2 to generalize reservoir merging to multiple input domains. A reservoir r stores a path X_r, its weight W_r, and a sample count M_r, as per the traditional use of reservoirs for sampling X_r from a stream of inputs. In that context, M_r is the number of samples the current X_r is resampled from, as X_r is randomly retained or replaced with the right probability at the encounter of each new input sample, and M_r is increased by one. A *reservoir merge* of reservoirs r_1 and r_2 builds a new reservoir r_m, with X_{r_m} resampled from X_{r_1} and X_{r_2} as if it were resampled from the concatenation of the input samples of r_1 and r_2, and M_{r_m} is simply $M_{r_1} + M_{r_2}$.

The interpretation of M_r as a sample count is too strict for ReSTIR: a reservoir merge simply resamples X_{r_m} with RIS from canonical samples X_{r_1} and X_{r_2}, with resampling MIS weights $m_{r_i}(y) = M_{r_i}/(M_{r_1} + M_{r_2})$. The meaning of M_r in this context is relative weight for the corresponding sample. Since we use the X_r for estimating an integral and the M_r define the relative weights of these samples, we refer to the M_r as *confidence weights*. In fact, ReSTIR even caps M_r to a constant M_c, limiting confidence on old samples, invalidating the old interpretation as a sample count.

Reservoir merging generalizes to weighted GRIS, with proper MIS weights (see Section 6.4.6), simply by multiplying the $\hat{p}$ and $\hat{p}_{\leftarrow}$ in the MIS formulas by the corresponding reservoir's M_r; the resampling result is stored in X_{r_m}, and $M_{r_m} = \min(M_c, \sum_j M_{r_j})$. This generalized form of reservoir merging is used in the next section.

6.5.3 ReSTIR As Chained GRIS

We rewrite the key aspects of the ReSTIR algorithm, i.e., Bitterli et al. [7, Algorithm 5], as a sequence of GRIS resampling steps; we will refer to the stages of this algorithm later:

Let Y_i^{t-1} be a resampled (or sampled) path for pixel i on frame $t-1$, stored for later reuse along with its unbiased contribution weight $W_{Y_i^{t-1}}$. For each frame t, in ReSTIR, we

1. *(Initial candidates)* Generate an independent sample X_i^t for each pixel i and evaluate its contribution weight $W_{X_i^t}$.

2. *(Temporal reuse)* Use GRIS to select Z_i by resampling between last frame's sample Y_i^{t-1} and new sample X_i^t. Pixel correspondences may be identified via motion vectors.

3. *(Spatial reuse)* Each pixel selects numerous random spatial neighbors j, and selects Y_i^t by resampling between Z_i and neighbor samples Z_j via GRIS. This step may be executed multiple times with the assignment $Z_i := Y_i^t$.

4. Estimate the pixel integral, $I_i^t \approx f_i(Y_i^t) W_{Y_i^t}$.

ReSTIR typically stores a reservoir for each pixel i. The new samples X_i^t are treated as reservoirs with $M_r = 1$, and are merged with the reservoir storing Y_i^{t-1}, accounting for the confidence weights. Spatial resampling works akin to a stochastic convolution, sequentially merging in reservoirs from random nearby pixels. The last sample Y_i^t is stored in that pixel's reservoir, and its confidence weight from the spatial reuse passes is used in the next frame's temporal resampling step. Occluded samples get weighted $w_i = 0$ to avoid selection.

Section B.5 discusses additional details related to performance and correctness.

6.5.4 Path Space Exploration via M-capping

The capping of M_r to a constant M_c (Section 6.5.2) is critical to ReSTIR: without limiting M_r, the relative weights of new samples exponentially approach zero, causing convergence to the wrong result as in Figure 6.2.

With M-capping, the relative weight of the temporally reused sample is approximately limited to at most $M_c/(M_c+1)$, which should intuitively fulfill the convergence constraints in Section 6.4.7.[10] The abovementioned constraint is a requirement for *input samples*; even if the constraint is fulfilled, the ReSTIR result itself still will not converge since the number of spatial input samples is not increased ($M \not\to \infty$). Instead, ReSTIR will explore the path space in such a way that its average over frames will now converge in a still scene: Assume that we hypothetically resample, with $\hat{p} = f_i$, for pixel i a path from one of past frames, i.e., $Y = Y_i^s$ where s is random. Its PDF now approaches $f_i/\|f_i\|$, the variance of its contribution converges to zero, and the said contribution is the mean of the ReSTIR frames, $f_i(Y)W_Y = \frac{\cancel{f_i(Y)}}{\cancel{\hat{p}(Y)}} \sum_{t=1}^{T} \frac{1}{T} \cdot f_i(Y_i^t) W_{Y_i^t}$.

This gives us an intriguing interpretation of ReSTIR: with *state* defined as one path X_i for each pixel, ReSTIR produces an unbiased, explorative non-Markovian chain

[10]This would lead to $b_k = (M_c/(M_c+1))^k$ with $b_k \to 0$, but an exact mathematical proof is hard due to correlations and complicated MIS weights.

whose PDF approximates f better with more input samples. Averaging images of this chain converges in a still scene. In real-time, we display single states of the unbiased chain, updated in time by sampling, shifting and resampling paths.

6.5.5 ReSTIR for Offline Rendering

Temporal path reuse reduces current frame variance but correlates samples temporally. This slows convergence if temporally accumulating in a progressive renderer. Instead, we propose rendering independent frames with spatial-only GRIS. This speeds convergence and should allow easier integration to existing systems.

This offline algorithm is a simple two-pass method. In a single iteration, the first pass performs one or more rounds of cross-pixel reuse with GRIS to resample canonical samples from other pixels. The second pass simply averages the produced images.

Proving convergence of the mean is now easy, despite the correlations in the spatial reuse passes: GRIS with proper MIS weights remains unbiased despite the correlations, and by Equation 6.40, the contributions of GRIS with a fixed number of spatial reuse passes remain bounded. Hence, averaging independently sampled frames converges.

Strictly speaking, the above convergence is true only for scalar functions f_i. In practice, f_i is vector-valued, and we use e.g., a grayscale $\hat{p}_i = |f_i|$, which guarantees convergence of a path $Y = Y_i^s$ sampled from a random frame s to the brightness $|f_i|$. Literally evaluating the unbiased contribution yields,

$$f_i(Y)W_Y = \frac{f_i(Y)}{|f_i(Y)|} \cdot \sum_{t=1}^{T} \frac{1}{T} \left|f_i(Y_i^t)\right| W_{Y_i^t}, \tag{6.46}$$

which may include color noise. However, in the offline context we assume budget for multiple samples, and thus recommend the explicit mean formula

$$\tilde{I}_i = \frac{1}{T} \sum_{t=1}^{T} f_i(Y_i^t) W_{Y_i^t} \tag{6.47}$$

as it removes color noise.

The convergence of $Y = Y_i^s$ in brightness opens potentially interesting future work: a random subset of the Y_i^t could be resampled and stored for each pixel to e.g., bootstrap the rendering of the next animation frame, or for re-rendering after material changes.

6.6 Designing Shift Mappings

Previous light transport techniques that manipulate and reuse paths (e.g., gradient-domain rendering) introduce various shift mappings to map paths between pixels. Generally, no single shift map is optimal and the best one depends on both scene properties as well as their computational efficiency on different hardware. In this section, we describe key properties of effective shift maps and introduce common building blocks for practical shift mappings.

We also describe a novel design principle for effective GPU-based shift mappings, how to choose the shifting strategy based on the sampled BSDF lobe, and new heuristics for avoiding noise.

6.6.1 Shift Mapping

A shift map T takes a path $\bar{\mathbf{x}}$ from pixel k and maps it to another path $\bar{\mathbf{y}} = T(\bar{\mathbf{x}})$ in pixel j. We call the original path $\bar{\mathbf{x}}$ the *base path*, and the shifted $\bar{\mathbf{y}}$ the *offset path*. Using the vertex parametrization by Veach [160], we define a generic shift map T from Ω_k to Ω_j as

$$T([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ...]) = [\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ...] \ . \tag{6.48}$$

Vertex $\mathbf{y}_0$ is normally specified on the sensor and $\mathbf{y}_1$ comes from tracing through pixel j, accounting for depth-of-field parameters.

When designing shift maps, the main freedom (and challenge) is designing a heuristic for vertices $\mathbf{y}_2$ and beyond so the shift approximately retains the path contribution, $f_k(T_k(\bar{\mathbf{x}})) \approx f_j(\bar{\mathbf{x}})$. Maximizing similarity of path contributions roughly equates to reusing (nearly) the same paths for nearby pixels, a common design heuristic. Figure 6.4 shows a hybrid shift mapping of random replay and reconnection as an example.

• **Local decisions.** A common strategy to find offset paths $\bar{\mathbf{y}}$ builds them sequentially, vertex-by-vertex, starting from $\mathbf{y}_1$ and analyzing local base and offset path geometry. For each i, the next offset vertex $\mathbf{y}_{i+1}$ is decided based on base path vertices $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, and $\mathbf{x}_{i+1}$ plus offset path vertices $\mathbf{y}_{i-1}$ and $\mathbf{y}_i$. For example, if vertices $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ and $\mathbf{y}_i$ have rough materials, a common strategy connects the base and offset paths by choosing $\mathbf{y}_{i+1} = \mathbf{x}_{i+1}$. Previous vertices $\mathbf{x}_{i-1}$ and $\mathbf{y}_{i-1}$ can also be used to perform half-vector copy [108].

• **Ensuring bijectivity.** Sequential construction of offset paths sometimes halts abruptly: e.g., in half-vector copy, local decisions can map a refraction into total internal reflection,

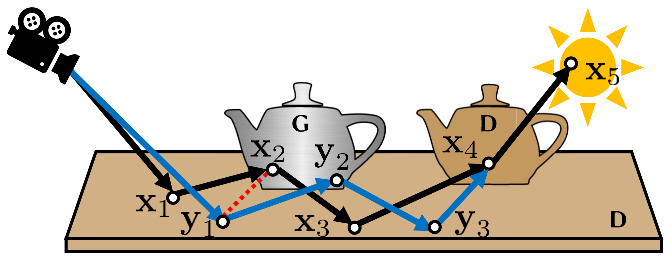


Figure 6.4: A hybrid shift mapping. The **base path** (black) selects $\mathbf{x}_4$ for reconnection, since both $\mathbf{x}_3$ and $\mathbf{x}_4$ are rough. The **offset path** (blue) copies the random numbers of the base at $\mathbf{x}_1$ and $\mathbf{x}_2$ to construct similar scatter directions for $\mathbf{y}_1$ and $\mathbf{y}_2$ and reconnects $\mathbf{y}_3$ to $\mathbf{x}_4$. This is the earliest reconnection giving two consecutive rough/diffuse vertices. Without connectability conditions the **offset path** (blue) would connect $\mathbf{y}_1$ to $\mathbf{x}_2$ (a glossy vertex), potentially giving a path of near-zero contribution as $\mathbf{y}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3$ is far from an ideal reflection.

but the reverse never happens, breaking bijectivity. Bijectivity is not always achievable throughout the path space, but that poses no big problem: not all paths need belong to the shift mapping domain. The shift may, after trying to shift a path, simply return "undefined." This marks the path as not belonging in the shift's domain.

Any successful shift must be invertible: if $\bar{\mathbf{x}}$ shifts to $\bar{\mathbf{y}}$, an inverse shift must exist to map $\bar{\mathbf{y}}$ back to $\bar{\mathbf{x}}$. Often slightly more is guaranteed by designing symmetric shift mappings where if $T_{k\to j}(\bar{\mathbf{x}}) = \bar{\mathbf{y}}$, then $T_{j\to k}(\bar{\mathbf{y}}) = \bar{\mathbf{x}}$. Removing paths from a map's domain may cause noise and waste computation, but neglecting bijectivity introduces significant bias.

6.6.2 Common Building Blocks

Local decisions for building offset paths often stem from fixing some property of the base path; such invariants share information between the paths, making shift inversion possible. Here we briefly review some common strategies for shift mapping:

- **Vertex copy (reconnection).** Reconnecting offset and base paths as soon as possible is common, as vertex sharing is cheap and often keeps path contributions similar. If $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, and $\mathbf{y}_i$ all lie on rough materials, Lehtinen et al. [107] reconnect the offset path to the base path by setting $\mathbf{y}_{i+1} = \mathbf{x}_{i+1}$. Subsequent vertices of $\bar{\mathbf{x}}$ are normally copied too. This strategy is good for diffuse and rough materials.

- **Half-vector copy.** Reconnection breaks path similarity for near-specular vertices. Kettunen et al. [108] transform the base path's half-vector into local tangent space, copy it to the offset path, and re-trace the vertex $\mathbf{y}_{i+1}$ in the reflection (or refraction) direction.

• **Direction copy.** Direction copy takes the exitant direction from a base path vertex and copies it to the offset path, in global coordinates, and re-traces to find the next vertex $\mathbf{y}_{i+1}$. Direction copy is often used with environment mapping.

• **Random replay.** Random replay copies the base path's random numbers to re-trace $\mathbf{y}_{i+1}$ with the method used by the base path. Random replay often makes decisions roughly similar to copying the half-vector or direction, or reconnecting to an area light in the case of next-event-estimation.

• **Manifold exploration.** If reconnection is impossible due to specularities, Lehtinen et al. [107] find the next connectable vertex, copy it, and apply manifold exploration [163] for intermediate (near-)specular vertices. This strategy iteratively constructs high-quality offset paths, albeit at relatively high cost.

While reconnecting quickly is often a good strategy, there are pitfalls. Some challenges for path shifts include: not closely approximating ideal reflections on high gloss surfaces, trying to reconnect through occlusions, shifting between different objects or materials, or simply diverging too far (e.g., due to reflection or refraction).

6.6.3 A Full Shift Mapping

Full shift mappings combine these building blocks, often based on simple heuristics. For example, Kettunen et al. [108] sequentially analyze base and offset paths to find suitable reconnections using a simple condition: vertices $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, and $\mathbf{y}_i$ must all be considered "sufficiently" rough. If this test passes, the base and offset paths are reconnected, otherwise a half-vector copy is used and the test is repeated for the next vertex. Hua et al. [114] show equivalent results by replacing the half-vector copy by random replay. We found the approach by Hua et al. more efficient on the GPU and slightly more general, so we adopt it with several improvements.

6.6.4 Shift Mappings Optimized for Real-Time Rendering

We study our ReSTIR PT with two different shift mappings, and modify them as needed to make them suitable for a GPU implementation that can target real-time rendering:

- The reconnection shift [107] always connects to the first indirect vertex by setting $\mathbf{y}_2 = \mathbf{x}_2$, which usually works well for sufficiently diffuse scenes. ReSTIR GI [10] also implicitly uses this choice but trades correctness for performance.

- A hybrid of random replay and reconnection [114] that postpones the reconnection by using random replay if certain connectability conditions are not fulfilled. We present an improved variant of this shift mapping.

The reconnection shift is easy to implement efficiently: it only requires storing the reconnection vertex and re-evaluating the path contribution. Implementing the hybrid shift efficiently is non-trivial: random replay is not efficient without reconnections, but due to potential reconnection postponing, all base path vertices need to be stored as candidates for reconnection. This is not ideal since GPU ray tracing is often memory-bound.

We propose minimizing memory use by letting the base path select a single potential reconnection vertex; we precompute the first base path vertex $\mathbf{x}_i$ that satisfies the connectability condition for $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. Reconnection must happen at this vertex, or it does not happen. This only requires storing vertex $\mathbf{x}_{i+1}$ instead of the full path.

This constraint is reasonable. For useful path reuse, the base and offset paths should be relatively similar; if similar enough, they should also agree on the reconnection index. Further, our bijectivity requirement *forces* this guarantee: when building $\mathbf{y}$, if we find it disagrees on the earliest possible reconnection vertex, the shift must return "undefined" as it would not be invertible.

6.6.5 Connectability Conditions

We propose two novel improvements for the connectability conditions compared to previous work, and we find these to often result in a significant noise and artifact reduction with our method.

• **Distance condition.** Area formulations of the rendering equation include geometry terms that become singular for short path segments, e.g., in corners. In unidirectional path tracing, this singularity stems from next-event-estimation but is eliminated by standard MIS. Similar singularities appear when reconnecting nearby vertices, causing increased noise near geometry edges.

We propose reducing this problem by skipping reconnections that introduce short segments. This is similar to distances test in Manzi et al.[162], but instead of spawning a manifold walk, we postpone reconnection by performing random replay. More concretely, we only allow reconnection to $\mathbf{x}_{i+1}$ if $\|\mathbf{x}_{i+1}-\mathbf{x}_i\| \geq d_{\max}$. By symmetry, the offset path must fulfill $\|\mathbf{x}_{i+1}-\mathbf{y}_i\| \geq d_{\max}$ for $\mathbf{y}_{i+1}$ to become $\mathbf{x}_{i+1}$, as described in Section 6.6.4.

• **Lobe-specific connectability.** Kettunen et al. [108] test reconnection feasibility by ensuring roughness values of $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, and $\mathbf{y}_i$ all exceed a given threshold. But since BSDFs often sum multiple separate lobes, such tests are ambiguous, e.g., on a material with a bottom diffuse layer and a top clear coat. Choosing good shift maps for such materials is a long-standing problem [164]: how to classify roughness of composite materials with just one parameter? Any single strategy likely mistreats at least one layer.

Unidirectional path tracers often optimize importance sampling by decomposing and evaluating only one BSDFs lobe per vertex [165]. We propose the same for shift mappings: we examine the roughness of just the selected lobe, and otherwise proceed as described in Section 6.6.4. We detail the path extension with lobe indices that is required for lobe-specific connectability in Section 6.6.6.

6.6.6 Extending Paths with Lobe Indices

For a path tracer generating N paths by different techniques for each path length d, the standard path integral can be written

$$I = \sum_{d=1}^{\infty} \sum_{n=1}^{N} \int_{\Omega_d} \omega_n(\bar{\mathbf{x}}) f(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}\,, \tag{6.49}$$

where ω_n is the MIS weight for path strategy n. For simplicity, we assume here that $N = 2$, and $n = 1$ uses next-event estimation to sample the last vertex and $n = 2$ uses BSDF sampling. Balance heuristic MIS weights give $\omega_n(\bar{\mathbf{x}}) = \frac{p_n(\bar{\mathbf{x}})}{p_1(\bar{\mathbf{x}})+p_2(\bar{\mathbf{x}})}$, where p_1 and p_2 are the NEE and BSDF sampling PDFs of the path, and sum over all BSDF lobes.

Our improved shift strategies from Section 6.6.4 require splitting BSDFs into lobes. We transform Equation 6.49 to use a lobe-extended path space, allowing us to implement ReSTIR by simple substitution, without any need for heuristic argumentation.

We combine paths $\bar{\mathbf{x}} = (\mathbf{x}_0, \ldots, \mathbf{x}_d)$ with sequences of lobe indices $\bar{\ell} = (\ell_1, \ldots, \ell_{d-1})$ into an extended path space of length-d paths $\tilde{\Omega}_d$ represented by pairs $(\bar{\mathbf{x}}, \bar{\ell})$. Each ℓ_j is a positive integer $1 \leq \ell_j \leq N_{\text{lobe}}$ (the BRDF model's lobe count) or special symbol $\ell_{d-1} = \mathcal{N}$ if our path ends with next-event estimation.

With f denoting the usual path contribution function, we define a partial contribution function $f_{\bar{\ell}}$ as follows: for fully BSDF-sampled paths, it evaluates only lobe ℓ_j at each vertex

$\mathbf{x}_j$. If the last vertex $\mathbf{x}_d$ is NEE-sampled, i.e., $\ell_{d-1} = \mathscr{N}$, the BSDF at $\mathbf{x}_{d-1}$ is evaluated with all lobes.

Denoting the set of lobe index sequences by L_d, we can rewrite Equation 6.49 as

$$I = \sum_{d=1}^{\infty} \sum_{\bar{\ell} \in L_d} \int_{\Omega_d} \omega_{n(\bar{\ell})}(\bar{\mathbf{x}}) f_{\bar{\ell}}(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}, \tag{6.50}$$

for $n(\bar{\ell}) \in \{1, 2\}$ based on whether the last vertex is NEE- or BSDF-sampled. Finally, we combine the sums into an integral similar to Veach [160], which integrates over our extended path space $\tilde{\mathbf{x}} \in \tilde{\Omega}$ for pairs $\tilde{\mathbf{x}} = (\bar{\mathbf{x}}, \bar{\ell})$ of all lengths d:

$$I = \int_{\tilde{\Omega}} \omega_{n(\bar{\ell})}(\bar{\mathbf{x}}) f_{\bar{\ell}}(\bar{\mathbf{x}})\, \mathrm{d}\tilde{\mathbf{x}}. \tag{6.51}$$

This formulation allows use of shift mappings that reason about the BSDF lobes, which is not possible in vertex-based path spaces.

To shift a path, we test if vertex $\mathbf{x}_j$ is sufficiently rough by examining the roughness of the lobe ℓ_j chosen to sample vertex $\mathbf{x}_{j+1}$. We treat NEE-sampled vertices as rough if at least *one* of their BRDF lobes is sufficiently rough. All light vertices are treated as rough. If all three reconnection vertices ($\mathbf{x}_j$, $\mathbf{x}_{j+1}$ and $\mathbf{y}_j$) pass the roughness and distance conditions (Section 6.6.5), we execute the reconnection; otherwise we sample $\mathbf{y}_{j+1}$ via random replay.

When using random replay, base and offset paths typically select the same BRDF lobes, as reusing random numbers over nearby paths gives similar per-vertex choices. A reconnection shift copies the lobe index from the base path vertex. In both cases, path contribution is likely preserved. Separating BRDF lobes frequently increases efficiency of our hybrid shift significantly.

6.7 Implementation

We apply our GRIS theory in a proof-of-concept path tracing algorithm we call ReSTIR path tracing (ReSTIR PT). We build on the Falcor GPU rendering framework [166], and implement ReSTIR PT as chained GRIS passes, per Section 6.5.3.

ReSTIR PT can use any shift map to reuse paths between pixels, but we implement the two from the previous section: a *hybrid shift* combining random replay and reconnection with our lobe-specific improvements, and a simpler *reconnection shift* that always reconnects to the first indirect vertex.

Like many path tracers, ours only evaluates the sampled BSDF lobe for BSDF-sampled vertices and evaluates all lobes for NEE-sampled vertices. We treat lobe selections as additional path parameters, as described in Section 6.6.6, using the sampled lobe roughness to choose between reconnection and random replay.

Our ReSTIR PT implementation handles full surface-to-surface light transport. Volumetric media requires a volumetric shift map; Lin et al. [157] implicitly defines one possibility and Gruson et al. [109] propose another, though finding fast volumetric shifts for resampling remains interesting future work.

We have two prototypes, targeting unbiased real-time and offline light transport, though neither is performance optimized. This contrasts with Ouyang et al. [10], a biased[11] but optimized precursor. Building on our generalized theory, ReSTIR PT is an unbiased global illumination method that better handles specular light transport, thanks to supporting arbitrary shift maps.

While we expect benefits to direct illumination from our GRIS theory, our implementations primarily address indirect light. We use ReSTIR DI [7] for direct lighting.

Below, we discuss our design choices and implementation details.

6.7.1 Jacobian Determinants

We first give Jacobian determinants for the reconnection and random replay shifts. We assume base and offset paths up to vertex i are fixed; probability densities below are to be understood as conditional to earlier path state.

We denote by ω_i^x the unit vector from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$, and the corresponding random numbers leading from vertex $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ by $\bar{\mathbf{u}}_i^x$. The offset path features similar notation with y.

• **Solid angle.** When using the common solid angle parametrization, the Jacobian for the reconnection shift is (e.g., Kettunen et al. [108])

$$\left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| = \left|\frac{\cos \theta_2^y}{\cos \theta_2^x}\right| \frac{|x_{i+1} - x_i|^2}{|x_{i+1} - y_i|^2}, \tag{6.52}$$

[11] Ouyang et al. [10, Section 4.3] explain some of their key sources of bias; following our GRIS theory removes the bias.

for $\theta_2^\bullet$ the angle between $\omega_i^\bullet$ and the geometric surface normal at $x_{i+1} = y_{i+1}$. The Jacobian for deciding y_{i+1} by random replay is

$$\left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| = \left|\frac{\partial \omega_i^y}{\partial \bar{\mathbf{u}}_i^y}\right| \left|\frac{\partial \bar{\mathbf{u}}_i^y}{\partial \bar{\mathbf{u}}_i^x}\right| \left|\frac{\partial \bar{\mathbf{u}}_i^x}{\partial \omega_i^x}\right| = \frac{p_{\omega_i^x}(x_{i+1})}{p_{\omega_i^y}(y_{i+1})}, \tag{6.53}$$

i.e., the ratio of the solid angle sampling probabilities of the next vertices, given the paths up to vertex i.

As we use local shift decisions, the Jacobian of the full path shift is the product of Jacobians from each vertex.

• **Primary-sample space (PSS).** Path tracers build paths $\bar{\mathbf{x}}$ based on random number sequences $\bar{\mathbf{u}} = (u_1, u_2, \ldots)$, defining a primary-sample space $\mathscr{U}$. In practice, integrals of form $\int_\Omega f(\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}}$ are evaluated $\int_{\mathscr{U}} \frac{f(x(\bar{\mathbf{u}}))}{p_X(x(\bar{\mathbf{u}}))}\,\mathrm{d}\bar{\mathbf{u}}$, where x builds paths using random numbers.

Our prototypes use primary-sample space for easier implementation. Results are identical, but interpretations change: the integrand is $f(x(\bar{\mathbf{u}}))/p_X(x(\bar{\mathbf{u}}))$ over domain $\mathscr{U}$ and the PDF of the primary samples is $p_U = 1$. Section B.6 briefly introduces the PSS formulation of the path integral.

The Jacobian for random replay in the PSS parametrization is always 1, and solid-angle Jacobians can be converted into primary-sample space by dividing by the right-hand-side of Equation 6.53. Due to the vertex-by-vertex construction of the path, we have

$$\left|\frac{\partial \bar{\mathbf{u}}_i^y}{\partial \bar{\mathbf{u}}_i^x}\right| = \left|\frac{\partial \bar{\mathbf{u}}_i^y}{\partial \omega_i^y}\right| \left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| \left|\frac{\partial \omega_i^x}{\partial \bar{\mathbf{u}}_x}\right| = \frac{p_{\omega_i^y}(y_{i+1})}{p_{\omega_i^x}(x_{i+1})} \left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| . \tag{6.54}$$

• **Mixing PSS and path space shifts ⋆.** Mixing random replay with path space shifts poses a conceptual challenge. Path samplers often consume more random numbers than the path dimensionality; this dimensionality mismatch means Jacobians between path space and primary-sample space do not exist. Bitterli et al. [167] bijectively map between paths and their random numbers by padding paths with extra, unused dimensions. Assuming this theoretical bijection exists often allows mixing path space shifts and random replay.

6.7.2 Reservoir Storage

Our reservoirs (Section 6.5.3) consume 88 bytes per path while supporting our hybrid shift. Beyond storing contribution weights W_r and confidence weights M_r (Section 6.5), we store information needed for our shift map: the path's chosen reconnection vertex and a seed for random replay.

We provide the details of our reservoir data structure in Algorithm 1. A key takeaway is that most of the storage is used for enabling a reconnection to the base path's vertex: reconnection requires evaluating offset path's visibility to the reconnection vertex and the BSDF towards base path's next vertex. Note that our reservoir data structure is unoptimized and highly compressible–real-time use would allow lossy compression for increased performance, but our prototype implementation does not do it.

6.7.3 Parameters

In the following, we refer to steps of the ReSTIR algorithm as described in Section 6.5.3.

• **Offline.** For offline rendering, we sample for each pixel 32 initial samples (path trees) with path tracing, and resample one path with RIS similarly to Lin et al. [157]. The selected path gets reused over three iterations of spatial reuse. All spatial reuse passes resample from the current pixel and six neighbors selected from a 10-pixel radius via a low-discrepancy sequence. We found this a near-optimal configuration for offline rendering. Section B.7 contains a parameter ablation.

• **Real-time.** For real-time rendering, we resample the path from only one path tree sampled with path tracing. A single spatial reuse pass selects three random neighbors in

Algorithm 1: Content of the reservoir struct (88 bytes).

```
struct Reservoir
    float M; // Confidence weight (for e.g., M-capping).
    float W; // Unbiased contribution weight.
    float3 F; // Cached integrand value of the sample.
    uint pathFlags; // Path length, technique type, reconn. vertex id, etc.
    uint initRandomSeed; // Random state at primary hit $\bar{x}_1$.
    // Information about the reconnection vertex (rc):
    uint rcVertexRandomSeed; // Random state at reconn. vertex.
    uint rcVertexInstanceID; // Hit point information:
    uint rcVertexPrimitiveIndex;
    float2 rcVertexBarycentrics;
    float3 rcVertexWi; // Direction to next vertex of base path.
    float3 rcVertexRadiance; // Incident radiance from next vertex.
    float4 rcVertexCachedValues; // Various partial terms for evaluating the Jacobian at
        reconnection. Light sampling PDF for the MIS weight is also stored here.
```

a 20-pixel radius; this keeps spatial reuse costs low and relies more on temporal reuse to improve distributions. We cap M with $M_c = 20$, i.e., the prior frame confidence is at most $20\times$ that of new samples.

• **Connectability thresholds.** Good values for distance and roughness thresholds are important, but relatively consistent. A per-lobe GGX-roughness threshold of 0.2 generally works well; scene scale affects the distance threshold, we used 1% - 5% scene size in different test scenes. The camera distance to the region of interest and material glossiness affects the optimal parameter. We leave automatically setting ideal parameters to future work.

• **Resampling MIS.** We use the defensive variant of pairwise MIS (Equation 6.38) for spatial reuse. The real-time variant additionally uses generalized Talbot MIS (Equation 6.36) for temporal reuse. For both, $|R| = 1$, and convergence is realized in the offline case by rendering multiple independent frames.

6.8 Results and Discussion

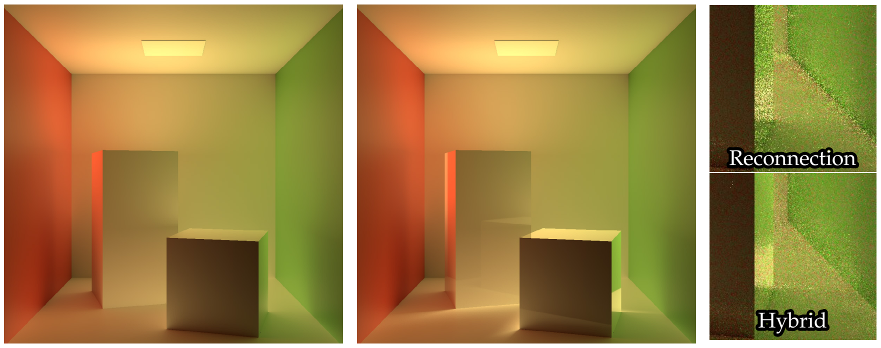
Below we first validate that convergence of ReSTIR PT matches our guarantees from Section 6.4. In Section 6.8.2 we quantify the quality of our shift maps from Section 6.6, and in Section 6.8.3, compare ReSTIR PT to recent global illumination algorithms. We present results separately for our real-time and offline variants. All performance numbers were measured on a GeForce RTX 3090 at 1920×1080.

6.8.1 Convergence Results

We study convergence in the *Cornell Box* scene, with modified materials mixing Lambertian and GGX microfacet BSDFs [168] with roughness 0.5 (Figure 6.5a).

We evaluate convergence behavior for three-bounce indirect illumination, and use the reconnection shift for simplicity. We compute error in grayscale images to map the convergence results to GRIS.

• **Asymptotic convergence with fixed reuse window.** Section 6.4.6 gives robust Talbot and pairwise resampling MIS weights that (Section 6.4.7) asymptotically realize single-sample zero-variance integration, given certain correlation and importance sampling criteria. In Figure 6.6, we show an experiment that fixes the ratio $|R|/M$ as we increase the



(a) Default *Cornell Box* (b) With glossy boxes (c) Two shifts

Figure 6.5: The reference rendering (3-bounce indirect lighting) of the *Cornell Box* scene for the two variants. (a) Default version, roughness 0.5. (b) Two glossy boxes with roughness 0.15. (c) 1 spp comparison of real-time rendering quality of ReSTIR PT with reconnection shift (5 ms) and hybrid shift (12 ms) in the glossy variant.

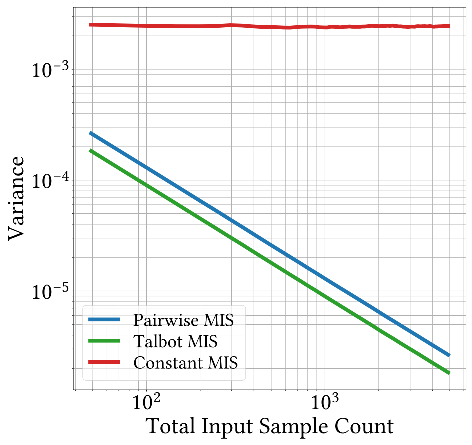


Figure 6.6: Single-sample integration error with GRIS, with more and more input samples from each pixel in a constant reuse window (7×7 pixels). As predicted, asymptotic zero-variance integration is realized with generalized Talbot and pairwise MIS weights but not with constant MIS weights (red). Results are the average of 10 independent executions.

independent input sample count over a fixed reuse window. This experiment uses only spatial reuse (no temporal).

Both our generalized pairwise and Talbot MIS weights realize a linear curve showing asymptotic zero-variance integration, but constant MIS weights do not. While Talbot weights have lower per-sample error, the lower algorithmic complexity of pairwise MIS can achieve similar variance 6-7$\times$ cheaper.

• **Non-convergence with increasing reuse window.** We only guarantee convergence if $|R| > O(\sqrt{M})$. A case breaking this condition takes one sample from each pixel in an increasing window, i.e., $|R| = 1$ but M grows. This does not guarantee convergence: some of path space is only covered by the canonical (central) sample. Adding only non-canonical samples decreases the chance to select the canonical one, so poorly-covered areas become importance sampled worse. Figure 6.7 shows how reusing over a larger window initially lowers variance, but beyond some point actually increases variance.

• **Temporal history and M-cap.** In Section 6.5.3 we discuss capping the temporal confidence weight M_r in ReSTIR reservoirs. As the cap M_c increases, b_k approaches 1 in Section 6.4.7. Using $M_c = \infty$ corresponds to $b_k = 1$, where we lose all guarantees. Figure 6.2 shows a simple example of this failure, converging to a static and wrong result.

In Figure 6.8a we show ReSTIR PT's integration error with temporal reuse, with increasing frame counts. Colors correspond to different M-cap values; the scene is static to avoid errors from animation.

Pixels compute a new independent sample on each iteration, which is resampled with the temporal result from the prior frame. The old sample's relative weight is $M_r/(M_r+1)$, which drastically favors this sample. This is akin to an exponential moving average.

Reusing prior frame samples improves Monte Carlo variance in the beginning, but until M_r reaches the cap, the relative weight of the new samples diminishes, increasing frame-to-frame correlation. This correlation buildup is analogous to *sample impoverishment*

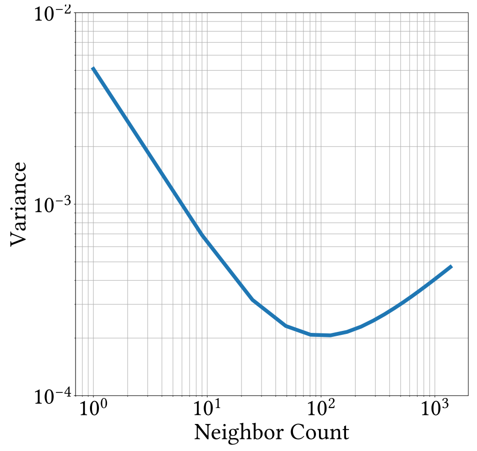
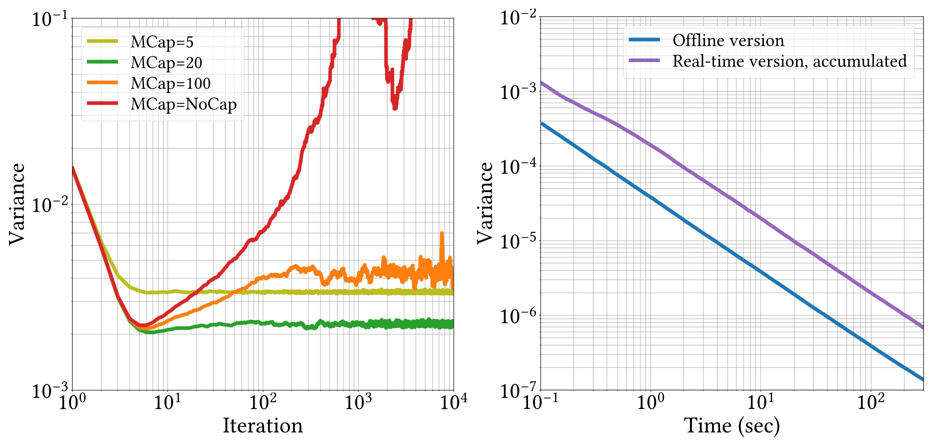


Figure 6.7: Breaking $|R| > O(\sqrt{M})$ loses convergence guarantees. Here, we reuse from a central (canonical) sample ($|R| = 1$) and an increasing window of M pixels around it. Even with proper MIS weights, without new canonical samples faraway pixels are increasingly worse matches. Eventually, this offsets any benefit from reusing more paths.



(a) Effect of M-cap (b) Real-time (M=20) vs. offline

Figure 6.8: Results showing different convergence properties of different parameters. (a) Error of ReSTIR PT with temporal reuse, with increasing frame counts and different M-cap values. A large M-cap eventually increases noise, while low values do not minimize error. Good M-caps (green) give consistently low errors. (b) Our offline method (blue) turns off temporal reuse, which converges faster when averaging frames; it avoids the frame-to-frame correlation introduced by temporal reuse.

in the SIR literature: fewer and fewer truly different samples remain, unless this correlation buildup is halted.

Eventually, increased correlation overshadows the benefits of reuse (red, orange) leading to higher variance until M-cap is reached. Capping M to a value that maximizes the benefits of temporal reuse (green) is foundational to real-time rendering with ReSTIR.

In Figure 6.8b we show the convergence behavior when averaging consecutive frames of our real-time ReSTIR PT with a finite M-cap (purple). We empirically find convergence, as the M-cap decorrelates temporally distant frames, i.e., it forgets temporal correlations.

• **Offline rendering.** The goal of offline rendering is slightly different: instead of maximizing the individual quality of each frame, we want to produce the best image over a longer rendering time. We find that for this purpose, the correlations from temporal reuse hurt more than they help, and we propose turning off temporal reuse for offline rendering. The additional rendering time allows us to use slightly different rendering parameters (Section 6.7.3), and due to both improvements, the resulting algorithm often converges significantly faster (Figure 6.8b, blue).

6.8.2 Shift Mapping Results

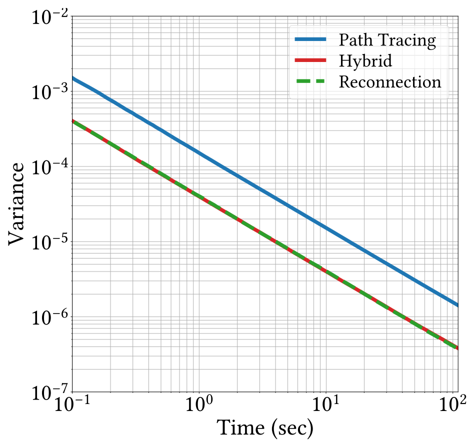
We study ReSTIR PT with the reconnection and hybrid shifts in the *Cornell Box* scene. We analyze the results in the rough variant (GGX roughness 0.5, Figure 6.5a), and variant with glossy boxes (roughness 0.15, Figure 6.5b).

• **Reconnection is good for rough.** We plot convergence in Figure 6.9 with a path tracing baseline. As known from gradient-domain rendering (e.g., Kettunen et al. [108]), the reconnection shift is efficient for rough surfaces (Figure 6.9a, green) allowing cheap path reuse from indirect light. Because the reconnection shift (green) always reconnects after a primary hit, regardless of BSDF, it less efficiently reuses paths involving glossy interactions (Figure 6.9b, green).

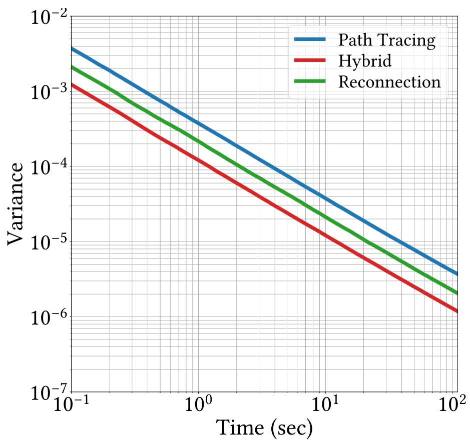
• **Hybrid is more robust.** Random replay better handles glossy surfaces; our hybrid shift inherits this property (Figure 6.9b, red) while also remaining effective on rough surfaces (Figure 6.9a, red).

• **Visual comparison.** Figure 6.5 shows shift map behavior on varied material types. For the glossy *Cornell Box* we render one path per pixel with ReSTIR PT (Figure 6.5c) with spatiotemporal reuse. On rough surfaces our hybrid shift (Figure 6.5c, bottom) behaves similar to a reconnection shift (Figure 6.5c, top), but our hybrid's distance criteria helps decrease noise at box edges.

Our hybrid shift postpones reconnections on glossy materials by inserting a random



(a) Default *Cornell Box*



(b) With glossy boxes

Figure 6.9: Error comparison of our ReSTIR PT with the reconnection shift (blue) and the hybrid shift (green). Both shift mappings are good for rough scenes (left), but the hybrid shift (green) is more suited for scenes with glossy surfaces (right).

replay. This often improves later reconnections, reducing noise on glossy surfaces (Figure 6.5c, side of glossy box).

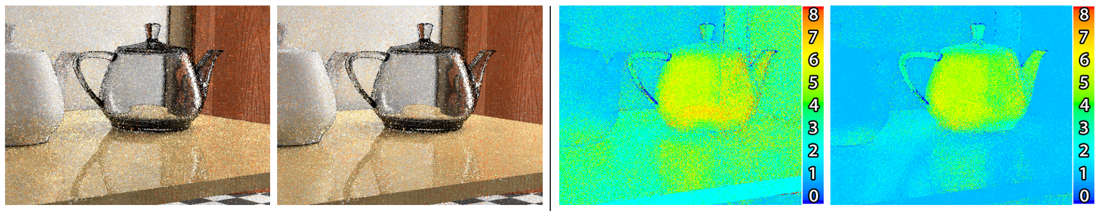
The lower reconnection shift quality is somewhat offset by its lower cost; equal-time comparisons allow averaging multiple independent iterations of the algorithm.

• **Separate handling of lobes.** Many renderers only evaluate one random lobe per BSDF evaluation. Varying the selected shift map per lobe significantly reduces noise and improves performance (see Figure 6.10). Reconnecting works well for rough BSDFs, and random replay is effective on glossy BSDFs. Neither shift is ideal for multi-lobe materials, but shifting lobes separately fixes the issue.

• **Caustics.** Caustics paths (i.e., {LS+DE} per Heckbert [169]) are important in highly specular scenes. Interestingly, we find our two shift maps work well for different types of caustics with ReSTIR PT.

Our hybrid shift effectively reuses paths for contact caustics, i.e., light concentrating on a nearby surface (see Figure 6.11, top). When tracing an offset path through the bunny, random replay produces paths similar to the base path; if this path hits the same light source, random replay gets good path reuse. Reconnection shifts, however, often fail to reconnect on near-delta BRDFs.

Caustics from distant highlights, e.g., the lamp reflection in Figure 6.11, bottom, perform poorly with the hybrid shift. Offset paths generated by random replay easily diverge enough to miss the small highlight, increasing noise. Conversely, reconnection only changes the incident direction slightly when reconnecting to the distant window, minimizing path divergence and increasing path contributions.



(a) All Lobes
MAPE: 0.316
Time: 51.7 ms

(b) Random Lobe
MAPE: 0.297
Time: 38.7 ms

(c) All Lobes
Avg. path length
pre-connection: 2.3

(d) Random Lobe
Avg. path length
pre-connection: 1.3

Figure 6.10: Evaluating just one random BSDF lobe enables lobe-specific shift maps providing more efficient reuse, less noise, and shorter render time. The heatmaps show lobe-specific shifts decrease the average path length on multi-lobe materials. Images are insets of the *VeachAjar* scene.

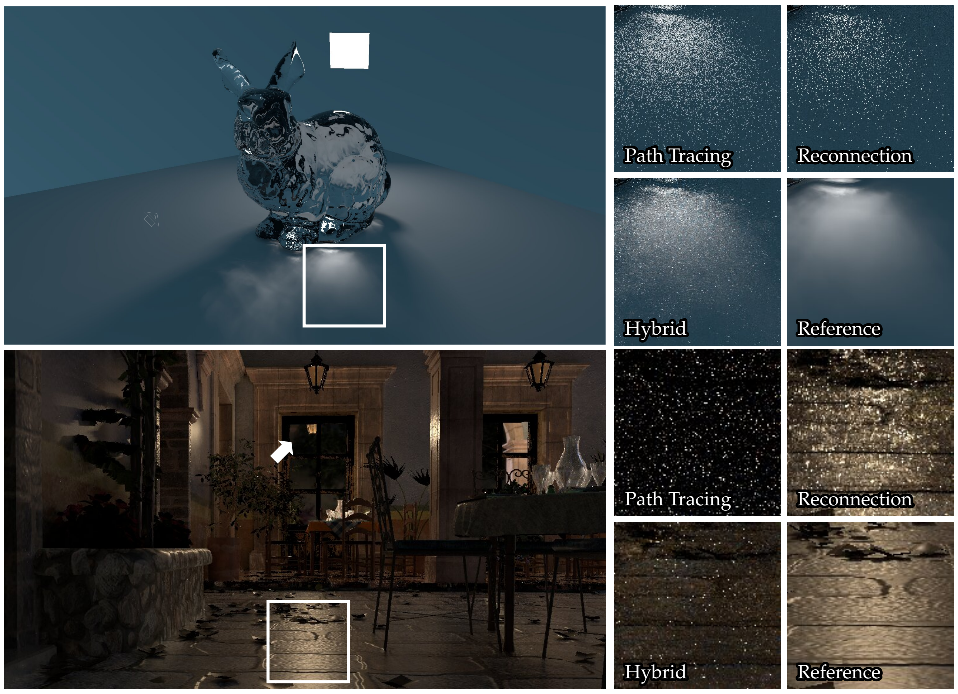


Figure 6.11: Top: Equal time (25 ms) comparison between path tracing and ReSTIR PT using two different shifts in a glass bunny scene. Bottom: Equal time (60 ms) comparison in the *SanMiguel* scene (showing only indirect lighting to avoid caustics made invisible by direct highlight). Notice how the two shift mappings are good for different kinds of caustics.

We expect a manifold exploration shift [107] would improve both cases; while expensive if applying to all paths, ReSTIR PT works with single path resampled from the path tree, making such a shift feasible. This is interesting future work.

6.8.3 Rendering Results

In this section we compare our results in three contexts: explicitly versus the path reuse by Bekaert et al. [11], general quality comparisons to other real-time methods, and comparisons in an offline context.

Average light levels vary widely across our scenes, so we tone map for visual presentation. But we report errors with MAPE[12] (mean absolute percentage error) on HDR results. This L1 metric is more resistant to occasional fireflies in sample-reuse algorithms.

For comparisons, we used Falcor's [166] built-in unidirectional path tracer, and implemented ReSTIR PT, Bekaert-style path reuse (BPR) [11], ReSTIR GI [10], and ReSTIR DI [7]

[12]We use $\text{MAPE}(I, I_{\text{gt}}) = \text{mean}\left(\frac{|I - I_{\text{gt}}|}{0.01 \cdot \text{mean}(\tilde{I}_{\text{gt}}) + \tilde{I}_{\text{gt}}}\right)$, for $\tilde{I}_{\text{gt}}$ a grayscale ground-truth.

using this framework.

6.8.3.1 Versus Path Reuse

In Figure 6.12, we compare ReSTIR PT with a reconnection shift, path tracing and BPR. We reimplemented BPR per Bekaert et al. [11], by sampling a path tree for each pixel, dividing the image into N-rooks tiles, and connecting pixels to all paths within a tile using a reconnection shift. We use 16-pixel tiles.

But path reuse over tiles, especially at low sample counts, introduces obvious tile boundaries. ReSTIR-style reuse does not introduce artificial edges, as reuse radii are selected separately per-pixel. Due to tile artifacts, we skip further real-time comparisons with Bekaert.

6.8.3.2 Real-Time Rendering

For real-time comparisons, *all* algorithms compute direct lighting via ReSTIR DI [7], unless otherwise mentioned. Image differences thus depend on how various algorithms compute indirect illumination. The second and third columns in Figure 6.13 toggle ReSTIR DI. It greatly reduces variance in direct light, but noise in indirect light requires other methods.

For equal-time comparisons, we render with ReSTIR PT using the hybrid shift and increase sample counts in other methods to reach (approximately) equal time. For ReSTIR PT with the reconnection shift, we run multiple independent ReSTIR chains simultaneously to match this cost. Figures 6.10 to 6.13 were captured during camera motion, to prevent

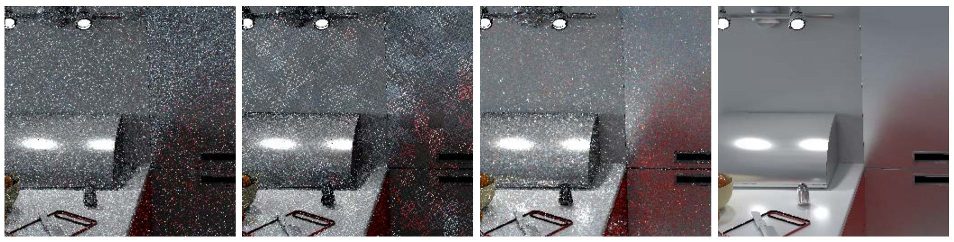

(a) Path Tracing MAPE: 0.958 (b) BPR MAPE: 0.898 (c) Ours (reconn.) MAPE: 0.325 (d) Reference

Figure 6.12: BPR [11] (b) often reduces error versus path tracing (a), but causes distracting structural artifacts at low sample counts. (c) Our ReSTIR PT gives less error without structural artifacts, despite also reusing spatially. Equal-time comparison in Kitchen (33 ms for uncropped images). All methods use ReSTIR DI for direct light.

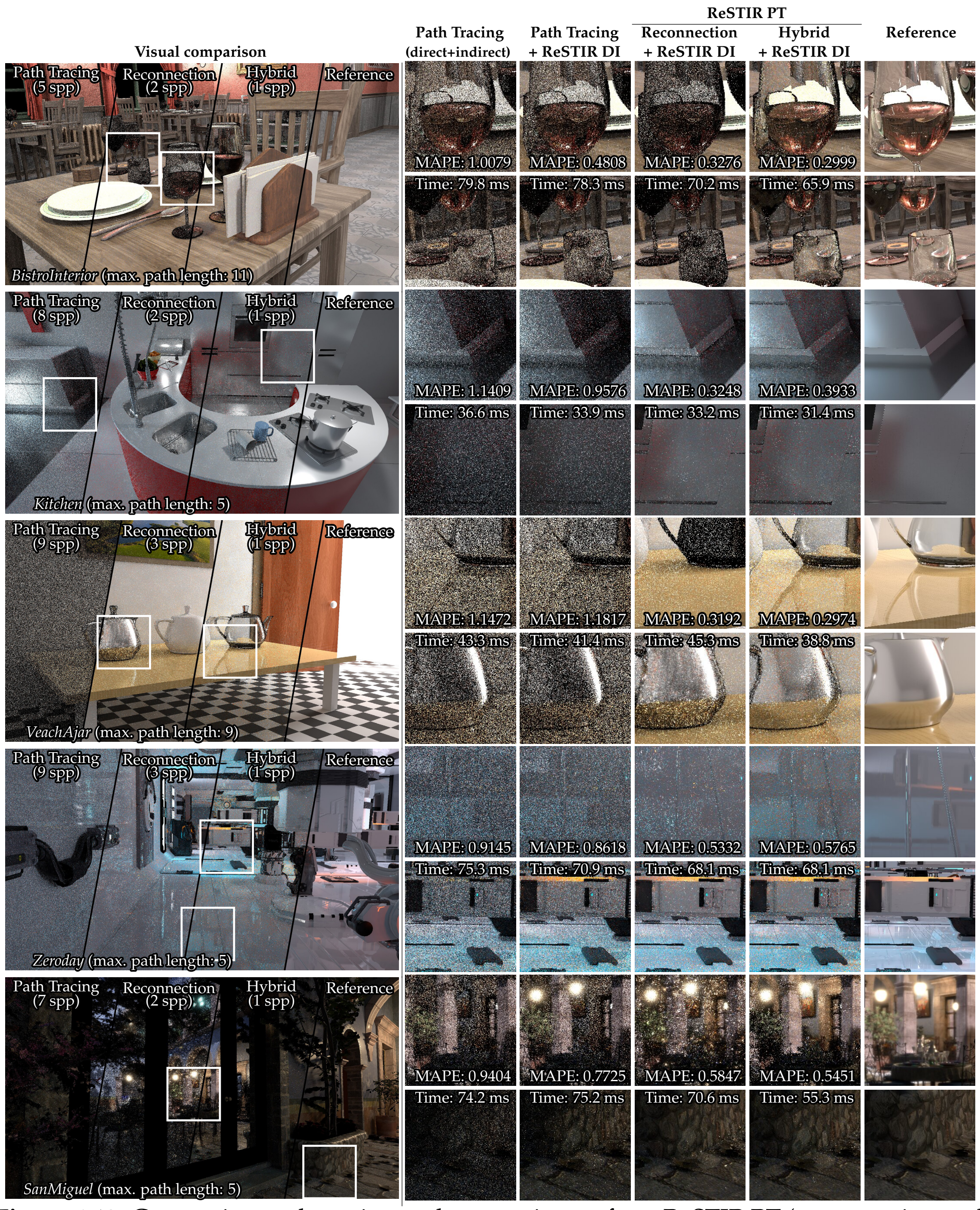


Figure 6.13: Comparing path tracing and two variants of our ReSTIR PT (reconnection and hybrid shift) for real-time rendering (all methods use ReSTIR DI for direct light). Images are captured during an animation sequence. MAPE are computed using HDR images.

ReSTIR PT from overly relying on temporal reuse for visual quality. We use no antialiasing or denoising.

In Figure 6.13, our ReSTIR PT with either our hybrid or reconnection shifts achieves the lowest error. Our hybrid shift significantly improves quality for glossy and refractive surfaces, e.g., the refractive wine glasses, the mirror in *SanMiguel*, the metal in VeachAjar, and the multi-layer surfaces in *ZeroDay* and *VeachAjar* scene.

Our hybrid shift also reduces noise near geometric edges (e.g., the first *Kitchen* inset). But the reconnection shift outperforms when more, cheaper samples are better, e.g., on sufficiently rough surfaces (the second *Kitchen* inset) or to help reduce color noise (the second Zeroday inset).

Numerically, ReSTIR PT achieves 24% - 75% lower MAPE compared to path tracing (both with ReSTIR DI), but we find subjective visual improvement much larger. The supplemental material contains a result viewer for visual inspection.

Figure 6.1 compares ReSTIR PT on two complex scenes, with MAPE 16% lower than ReSTIR GI. While ReSTIR GI handles diffuse surfaces well, glossy surfaces and refractions and handled poorly, introducing bias. All methods in Figure 6.13 are unbiased.

Figure 6.14 compares the denoised results. Without sufficient input sample quality, the appearance after denoising appears blurry. Our method significantly improves the sharpness compared to the baseline, especially for specular reflections/refractions.

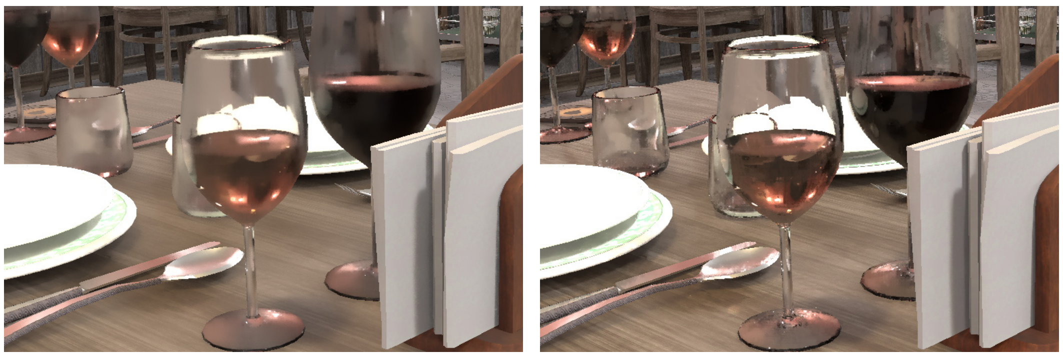

Baseline (5 spp) with denoising
ReSTIR PT (hybrid) (1 spp) with denoising

Figure 6.14: The NRD denoiser [12] applied to both path tracing and our method using the setup in Figure 6.13 but with a static camera (the denoiser cannot handle specular motion vectors, lowering the quality with camera motion).

6.8.3.3 Offline Rendering

For offline comparisons, we show 5 second equal-time renderings in representative scenes (see Figure 6.15). In Figure 6.16 we show convergence plots (up to 640 seconds) for all scenes. These figures show results and measure error *only* for indirect light.

Our ReSTIR PT with either hybrid or reconnection shifts outperforms path tracing and BPR in both short and long rendering times. BPR outperforms path tracing except in Zeroday, where BPR's reconnection shifts interact poorly with the shiny multi-layer surfaces. ReSTIR PT with reconnection shifts benefits from the amortization of sample costs when using GRIS, converging much faster than BPR.

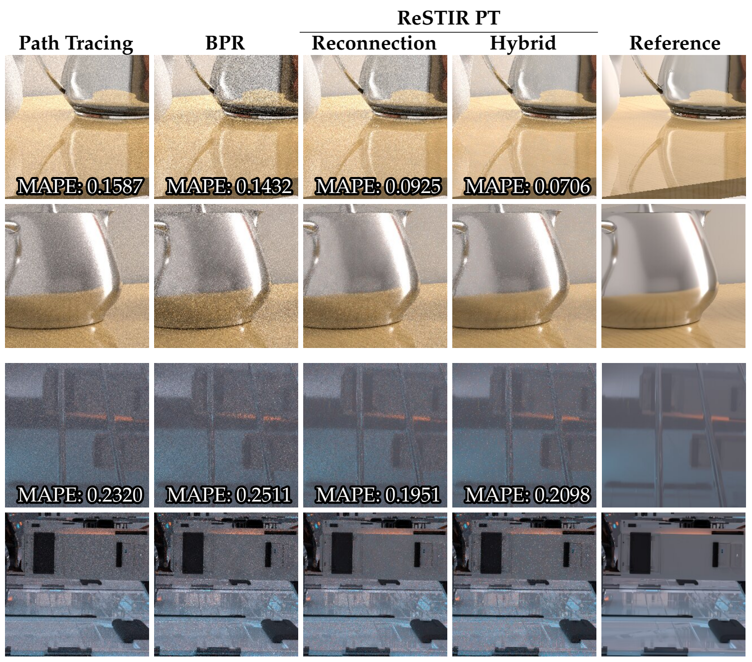


Figure 6.15: Visual comparison in the VeachAjar (first two rows) and Zeroday scenes using our offline ReSTIR variant, all rendered in 5 seconds.

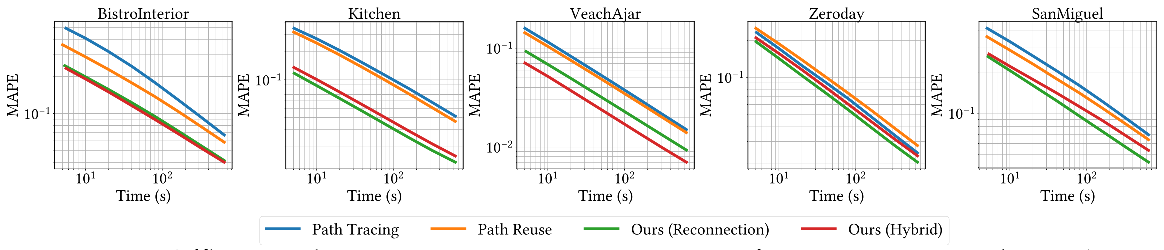


Figure 6.16: Offline rendering error comparison in terms of time spent on rendering (5-640 second range).

Across our scenes, BPR reaches the same MAPE up to 2.8$\times$ faster than path tracing, while our ReSTIR PT with the reconnection shift converges up to 14.4$\times$ faster, and ReSTIR PT with the hybrid shift converges up to 10$\times$ faster.

Interestingly, we find our hybrid shift's advantage in real-time rendering does not fully transfer to offline, though both provide a large improvement over the alternate methods. The only exception is *VeachAjar*, whose glossy teapots significantly benefit from the hybrid shift mapping in equal-time comparisons.

We postulate the following explanation. While better shift maps improve spatial reuse, they also improve temporal reuse (which improves *future* spatial reuse). Better temporal reuse is thus a very beneficial investment. For offline rendering we disable temporal reuse, losing this extra advantage from improved temporal shift maps (like our hybrid shift).

Equal-time comparisons are impacted by GPU thread divergence, as threads often trace path of different lengths. Comparing convergence at equal sample counts (Figure 6.17) rather than equal time, our hybrid shift is generally better than the reconnection shift, except for caustic paths in Figure 6.11 (bottom). This suggests run-time cost is overcoming our hybrid shift benefits. Divergence can be improved by better workload balancing and other optimizations, which may make our hybrid shift more generally appealing.

6.9 Conclusion

We introduce a new generalized RIS (GRIS) theory, which extends resampled importance sampling [8], to enable reusing correlated paths, taken from multiple domains (pixels) using context-aware shift mappings. Resampling gives asymptotically perfect importance sampling according to a user-specified target function $\hat{p}$; choosing $\hat{p} = f$ yields asymptotically zero-variance integration. See Figure 6.3 for a summary of key algorithmic differences from RIS.

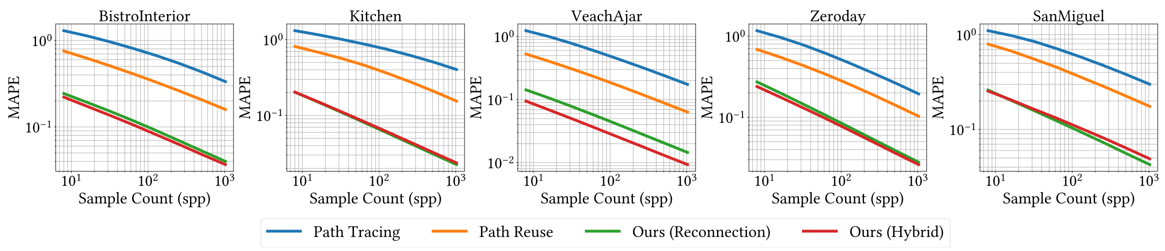


Figure 6.17: Offline rendering error comparison in terms of sample count.

With this theory as a foundation, we reformulate the spatiotemporal reuse in ReSTIR. We achieve unbiased path reuse that remains consistent even for long paths and complex specular light transport. As in Bitterli et al. [7] and Ouyang et al. [10], our streaming algorithms amortizes path generation over many pixels, leveraging GPU parallelism and temporal path histories to dramatically reduce variance at interactive framerates.

Beyond better robustness from a consistent and unbiased algorithm, we further improve on ReSTIR GI [10] by defining and using a new hybrid shift map to reduce noise. This extends the toolbox of commonly-used building blocks for shift maps by accounting for the sampled BSDF lobe and preselecting connection vertices to limit memory traffic. Somewhat surprisingly, ReSTIR PT can even reuse simple caustic paths in many cases.

Our theory provides conditions under which we can guarantee these improvements. Users must carefully account for domain changes using appropriate MIS weights; avoid unbounded $f/\hat{p}$ ratios; control resampling weight variance $\mathrm{Var}\left[\sum w_i\right]$; ensure an appropriate number of canonical paths to avoid undersampling parts of the domain; and, (if accumulating frames) use a reasonable M-cap to limit temporal correlation.

Based on these design constraints, we present an interactive ReSTIR PT but also show how to properly reuse paths spatially for consistent offline rendering. These redesigned algorithms avoid unpleasant surprises, e.g., cases like Figure 6.2 where pixels may converge slowly, if at all.

6.9.1 Future work

We believe our GRIS theory will help drive a variety of future research on resampling and path reuse algorithms.

• **Shift mappings and gradient-domain rendering.** Our extended lobe-path space could benefit shift mapping in gradient-domain problems, including for more efficient manifold exploration [107]. Manifold exploration could also help where local shift decisions prove limiting to our hybrid shift (e.g., Figure 6.11, bottom).

• **Color noise.** ReSTIR resamples paths using grayscale target functions $|f|$, which importance samples pixel brightness but not chroma $f/|f|$. Modifications could perhaps resample between wavelengths or use hero wavelengths to help with color noise.

• **MIS between shift mappings.** GRIS can naturally MIS between multiple shift maps, as it supports correlated samples. E.g., we could resample a single sample with multiple shift maps applied, automatically giving more effective shifts higher selection probabilities.

• **Undersampling.** GRIS, ReSTIR, and sample reuse can still suffer from undersampling. In screen space implementations, like ours, reuse may be limited by undersampling of tiny geometry or very high frequency light, e.g., sharp caustics. This can cause noise, streaks, or splotches and requires further investigation.

CHAPTER 7

CONCLUSION

In conclusion, we have presented a series of high-quality sampling methods that target different complex effects and show significant improvement over naive sampling and prior work, which justify our dissertation statement that high-quality sampling is important for rendering complex effects in real-time. More specifically:

- We have presented an extension of the lighting grid hierarchy method for efficiently computing high-quality diffuse-dominant global illumination at real-time frame rates, with dynamic scenes and dynamic lighting [170]. We introduce a low variance, biased sampling mechanism that samples the shadow term from a large number of virtual point lights, significantly improving the realism compared to prior work.

- We have presented a real-time light sampling technique for scenes with many lights, based on a perfect binary light tree and other extensions of stochastic lightcuts [171]. Our method minimizes the cost of light sampling, allowing more light samples within the same render time for achieving higher sampling quality. Our method significantly outperforms prior work in direct lighting of dynamic scenes with a large number of lights.

- We have presented a sampling solution for real-time volumetric rendering using spatiotemporal reservoir resampling [157]. Our solution extends resampled importance sampling and ReSTIR to path space, enabling interactive rendering of heterogeneous volumes in complex lighting environments. We show how biased transmittance estimates of increasing fidelity can be injected over multiple resampling steps to produce high-quality, unbiased results at a low cost. We demonstrate an efficient GPU implementation that renders effects that traditional real-time rendering methods fail to handle, like multiple scattering, while outperforming state-of-the-art sampling-based methods.

- Finally, we have presented a new generalized RIS (GRIS) theory and developed an algorithm, ReSTIR PT, that provides high-quality real-time rendering of general global illumination that contains long paths and complex specular light transport. Compared to RIS, our theory enables reusing correlated paths taken from multiple domains (pixels) using context-aware shift mappings. With this theory as a foundation, we can analyze the unbiasedness, consistency, and convergence behavior in existing ReSTIR methods and reformulate the spatiotemporal reuse in ReSTIR to derive a fully consistent and unbiased algorithm. Beyond achieving better robustness in real-time rendering, our new algorithm further improves ReSTIR-based path reuse (e.g., [10]) by defining a new hybrid shift map that accounts for the sampled BSDF lobe and preselects connection vertices to limit memory traffic. This leads to a significant sampling quality improvement compared to path tracing and prior work in path reuse.

7.1 Outlook

The sampling methods introduced in this dissertation are all somewhat based on reuse: we reuse lighting approximation in Chapter 3, importance sampling tree structure in Chapter 4, and sampling distribution in Chapter 5 and Chapter 6. Reuse always has a chance to fail in the presence of discontinuities. To make these algorithms more robust, reuse can be made smarter in a data-driven way. Alternatively, future work may improve the sampling quality before reuse. More advanced sampling algorithms including bidirectional path tracing [172, 173, 174, 175], path guiding [176, 177], path perturbations and mutations [178, 179], and even online machine learning [177, 156, 180, 181] can be combined with methods proposed in this dissertation to improve their robustness.

In addition, our methods all rely on a certain extent of denoising to produce final results. Our methods, especially those in Chapter 5 and Chapter 6, reuse information across pixels to improve sampling quality, which is equivalent to filtering probability distributions, introducing correlations in the noise. This correlation in the input signal is not handled well by many existing denoisers which are crafted for i.i.d. samples produced by path tracing. How to design denoisers that work with the correlated samples produced by our methods is an interesting future work.

While improving sampling algorithms is important, having hardware acceleration is often a crucial factor to the success of algorithms in productization. The importance of hardware support for real-time rendering has been repeatedly demonstrated in recent years from hardware tessellation for displacement mapping [182], RT core for real-time ray tracing [13], and Tensor core for deep-learning based denoising [14]. The research done by me and my collaborators [183, 184, 185, 186, 187] has also shown the importance of hardware acceleration in improving the speed and quality of real-time rendering. Even relatively brute-force sampling can be made practical for productization with hardware support [188]. So, redesigning or augmenting rendering hardware with the ability to overcome the bottlenecks in more advanced algorithms like the ones we introduced will be an important future work for these advanced sampling algorithms to fully achieve their theoretical capability in practical use.

APPENDIX A

RIS FOR PATH INTEGRALS

This appendix provides a more rigorous derivation of the RIS estimator of the path integral proposed in Section 5.3.3 in Chapter 5. We assume the path tracing process being a random walk that generates n paths $\boldsymbol{\lambda}^i (i \in \{1, ..., n\})$ that differ by lengths (or emission/scatter type in the volumetric case), responsible for n non-overlapping parts ($\int_{\boldsymbol{\Lambda}^i} F(\boldsymbol{\lambda}^i)\, d\boldsymbol{\lambda}^i$) of the path integral $L = \int_{\boldsymbol{\Lambda}} F(\boldsymbol{\lambda})\, d\boldsymbol{\lambda}$ (where $\boldsymbol{\lambda}$ can be any length or type). In other words, $L = \sum_{i=1}^{\infty} \int_{\boldsymbol{\Lambda}^i} F(\boldsymbol{\lambda}^i)\, d\boldsymbol{\lambda}^i$. An ordinary 1-sample Monte Carlo estimator simply sums the contributions from all sampled paths, i.e. $\langle L \rangle_{\text{MC}} = \sum_{i=1}^{n} \frac{F(\boldsymbol{\lambda}^i)}{p(\boldsymbol{\lambda}^i)}$. Note that n is a random variable and can pick any value from 0 to ∞. We know that $\langle L \rangle_{\text{MC}}$ is an unbiased estimator of the path integral L. However, we want to take advantage of RIS to avoid evaluating F for all paths. The traditional form of RIS assumes that all candidate samples are taken from the same sampling domain to estimate the same integral. Now, we want to use RIS to estimate a sum of integrals using candidate samples taken from the sampling domain of each integral, and only evaluate F for the chosen sample $\boldsymbol{\lambda}^r$. This requires a slightly different definition of RIS (the "$1/M$" term is removed from the estimator). We now show that the RIS estimator

$$\langle L \rangle_{\text{ris}} = \frac{F(\boldsymbol{\lambda}^r)}{\hat{p}(\boldsymbol{\lambda}^r)} \sum_{i=1}^{n} \frac{\hat{p}(\boldsymbol{\lambda}^i)}{p(\boldsymbol{\lambda}^i)} \tag{A.1}$$

where $\boldsymbol{\lambda}^r$ is chosen from $\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^n$ according to the weight $w(\boldsymbol{\lambda}^i) = \frac{\hat{p}(\boldsymbol{\lambda}^i)}{p(\boldsymbol{\lambda}^i)}$ is also an unbiased estimator of the path integral, i.e. $\mathbb{E}[\langle L \rangle_{\text{ris}}] = \mathbb{E}[\langle L \rangle_{\text{MC}}]$. The proof goes as follows,

$$\mathbb{E}\left[\langle L\rangle_{\text{ris}}\right] = \mathbb{E}\left[\mathbb{E}\left[\frac{F(\boldsymbol{\lambda}^r)}{\hat{p}(\boldsymbol{\lambda}^r)}|\boldsymbol{\lambda}^1,...,\boldsymbol{\lambda}^n\right]\cdot\sum_{i=1}^{n} w(\boldsymbol{\lambda}^i)\right] \tag{A.2}$$

$$= \mathbb{E}\left[\sum_{j=1}^{n}\left(\frac{F(\boldsymbol{\lambda}^j)}{\hat{p}(\boldsymbol{\lambda}^j)}\cdot\frac{w(\boldsymbol{\lambda}^j)}{\sum_{i=1}^{n} w(\boldsymbol{\lambda}^i)}\right)\cdot\sum_{i=1}^{n} w(\boldsymbol{\lambda}^i)\right] \tag{A.3}$$

$$= \mathbb{E}\left[\sum_{j=1}^{n}\frac{F(\boldsymbol{\lambda}^j)}{\hat{p}(\boldsymbol{\lambda}^j)}\cdot\frac{\hat{p}(\boldsymbol{\lambda}^j)}{p(\boldsymbol{\lambda}^j)}\right] \tag{A.4}$$

$$= \mathbb{E}\left[\sum_{j=1}^{n}\frac{F(\boldsymbol{\lambda}^j)}{p(\boldsymbol{\lambda}^j)}\right] = \mathbb{E}[\langle I\rangle_{MC}]. \quad \square \tag{A.5}$$

APPENDIX B

GENERALIZED RESAMPLED IMPORTANCE SAMPLING NOTES

This appendix includes theorems, proofs, derivations, notes, and additional experiments for generalized resampled importance sampling (Chapter 6).

B.1 Theorems

In this section, we present the most important mathematical theorems for generalized RIS with their proofs. A proof of Theorem 1 in Section 6.4.3 is also provided here.

B.1.1 Unbiased Contribution Weights, Section 6.3.2

Theorem A.1. *Let X and real-valued W be random variables in Ω. The following are equivalent:*

1. *For all integrable $f : \Omega \to \mathbb{R}$,*

$$\mathbb{E}\left[f(X)W\right] = \int_{\mathrm{supp}(X)} f(x)\,\mathrm{d}x, \tag{B.1}$$

2. *W is an unbiased estimator for X's reciprocal marginal density,*

$$\mathbb{E}\left[W \mid X\right] = \frac{1}{p_X(X)}. \tag{B.2}$$

Proof. Assume Item 1: W and X are such that $\mathbb{E}[f(X)W] = \int_{\mathrm{supp}\,X} f(x)\,\mathrm{d}x$ for any integrable f. We prove Item 2: Let $A \subset \mathrm{supp}(X)$ be measurable. Then,

$$\begin{aligned}
&\int_A p_X(x)\,\mathrm{d}x = \int_{\mathrm{supp}(X)} \mathbb{1}_A(x) p_X(x)\,\mathrm{d}x \\
&= \mathbb{E}\left[\mathbb{1}_A(X)p_X(X)W\right] = \mathbb{E}\left[\mathbb{E}\left[\mathbb{1}_A(X)p_X(X)W \mid X\right]\right] \\
&= \mathbb{E}\left[\mathbb{1}_A(X)p_X(X)\,\mathbb{E}\left[W \mid X\right]\right] \\
&= \int_A p_X(x)^2\,\mathbb{E}\left[W \mid X = x\right]\mathrm{d}x.
\end{aligned} \tag{B.3}$$

Since this holds for all measurable $A \subset \mathrm{supp}(X)$, we must have, almost everywhere in $\mathrm{supp}(X)$,

$$p_X(x) = p_X(x)^2\,\mathbb{E}\left[W \mid X = x\right]. \tag{B.4}$$

Since $p_X(x) > 0$ in $\operatorname{supp}(X)$, we deduce $\mathbb{E}[W \mid X = x] = 1/p_X(x)$ a.e. in $\operatorname{supp}(X)$.

Next, assume Item 2: W and X are such that $\mathbb{E}[W \mid X = x] = 1/p_X(x)$ a.e. in $\operatorname{supp}(X)$. We prove Item 1: Let $f : \Omega \to \mathbb{R}$ be integrable. We have

$$\begin{aligned} \mathbb{E}[f(X)W] &= \mathbb{E}[\mathbb{E}[f(X)W \mid X]] = \mathbb{E}[f(X)\,\mathbb{E}[W \mid X]] \\ &= \mathbb{E}\left[\frac{f(X)}{p_X(X)}\right] = \int_{\operatorname{supp}(X)} f(x)\,\mathrm{d}x. \end{aligned} \tag{B.5}$$

$\square$

B.1.2 Asymptotic Sample Distribution, Section 6.3.4

Theorem A.2 (Asymptotic Sample Distribution). *Assume, for each M (separately, starting from some M_0) a sequence of samples $(X_i \in \Omega_i)_{i=1}^M$ (we omit the index M for brevity), and that we resample Y_M ($=T_{s_M}(X_{s_M})$) proportionally to weights $w_{M,i}$ given by Equation 6.19. Assume also that the generated samples cover the support of $\hat{p}$,*

$$\operatorname{supp} \hat{p} \subset \operatorname{supp} Y_M \qquad \text{when } M \geq M_0. \tag{B.6}$$

If the variance of the weight sums tends to zero, i.e.,

$$\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i}\right] \xrightarrow{M\to\infty} 0, \tag{B.7}$$

Then,

1. *p_Y converges to $\bar{p}$ in probability, i.e., for any $\varepsilon > 0$*

$$\Pr[|p_Y(Y) - \bar{p}(Y)| > \varepsilon] \xrightarrow{M\to\infty} 0. \tag{B.8}$$

2. *the density ratio $\bar{p}(Y)/p_Y(Y)$ approaches 1 in mean square, i.e.,*

$$\mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right|^2\right] \xrightarrow{M\to\infty} 0. \tag{B.9}$$

3. *the integral of the absolute error of p_Y from $\bar{p}$ approaches 0, i.e.,*

$$\int_\Omega |p_Y(y) - \bar{p}(y)|\,\mathrm{d}y \xrightarrow{M\to\infty} 0. \tag{B.10}$$

4. *in the set in which $p_Y(y)$ converges, it converges to $\bar{p}(y)$ (except for a possible set of zero measure).*

5. *each subset of Ω will asymptotically get the correct ratio of samples.*

Proof. We start from Equation 6.22 and derive the equality

$$\sum_{i=1}^{M} w_i = \hat{p}(Y)W_Y. \tag{B.11}$$

We then prove the following, slightly more general result: If $(Y_M)_{M=M_0}^{\infty}$ is a sequence of random variables that fulfill $\operatorname{supp}\hat{p} \subset \operatorname{supp} Y_M$ (Equation B.6) and have non-negative unbiased contribution weights W_{Y_M} such that

$$\operatorname{Var}\left[\hat{p}(Y_M)W_{Y_M}\right] \xrightarrow{M\to\infty} 0 \tag{B.12}$$

(a generalization of Equation B.7), then the conclusions of Theorem A.2 hold:

Proof of Theorem A.2 (Item 1), Equation B.8. Let $\varepsilon > 0$ be given. We prove that

$$\Pr[|p_Y(Y) - \bar{p}(Y)| > \varepsilon] \xrightarrow{M\to\infty} 0 :$$

For any $0 < \varepsilon_2 < 1$ we have

$$\Pr\left[|\bar{p}(Y) - p_Y(Y)| > \varepsilon\right] = A + B + C, \quad \text{where} \tag{B.13}$$

$$\begin{aligned}
A &= \Pr\left[\Big(|\bar{p}(Y) - p_Y(Y)| > \varepsilon\Big) \wedge \left(p_Y(Y) \geq \frac{1}{\varepsilon_2}\right)\right], \\
B &= \Pr\left[\Big(|\bar{p}(Y) - p_Y(Y)| > \varepsilon\Big) \wedge \left(1 < p_Y(Y) < \frac{1}{\varepsilon_2}\right)\right], \\
C &= \Pr\left[\Big(|\bar{p}(Y) - p_Y(Y)| > \varepsilon\Big) \wedge \Big(p_Y(Y) \leq 1\Big)\right].
\end{aligned}$$

Since $\Pr\left[X \wedge Y\right] \leq \Pr\left[Y\right]$ and p_Y integrates to 1, we have

$$A \leq \Pr\left[p_Y(Y) \geq \frac{1}{\varepsilon_2}\right] \leq \varepsilon_2$$

(otherwise p_Y would integrate to more than $\varepsilon_2 \cdot 1/\varepsilon_2 = 1$). Since in case B we have $1/p_Y(Y) > \varepsilon_2$, we get

$$\begin{aligned}
B &= \Pr\left[\Big(|\bar{p}(Y) - p_Y(Y)|\,\varepsilon_2 > \varepsilon \cdot \varepsilon_2\Big) \wedge \left(1 < p_Y(Y) < \frac{1}{\varepsilon_2}\right)\right] \\
&\leq \Pr\left[\left(\frac{|\bar{p}(Y) - p_Y(Y)|}{p_Y(Y)} > \varepsilon \cdot \varepsilon_2\right) \wedge \left(1 < p_Y(Y) < \frac{1}{\varepsilon_2}\right)\right] \\
&\leq \Pr\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right| > \varepsilon \cdot \varepsilon_2\right].
\end{aligned}$$

Similarly, in case C we have $1/p_Y(Y) \geq 1$, and thus

$$\begin{aligned} C &= \Pr\left[\Big(|\bar{p}(Y) - p_Y(Y)| > \varepsilon\Big) \wedge \Big(p_Y(Y) \leq 1\Big)\right] \\ &\leq \Pr\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right| > \varepsilon\right]. \end{aligned}$$

By Chebyshev's inequality, we have for any $s > 0$ (such as $s = \varepsilon \cdot \varepsilon_2$ for B and $s = \epsilon$ for C), that

$$\Pr\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right| > s\right] < \frac{1}{s^2}\,\mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right|^2\right] \xrightarrow{M\to\infty} 0, \tag{B.14}$$

and thus

$$0 \leq \lim_{M\to\infty} A + B + C \leq \varepsilon_2 + 0 + 0 \xrightarrow{\varepsilon_2\to 0} 0, \tag{B.15}$$

i.e.,

$$\Pr\left[|\bar{p}(Y) - p_Y(Y)| > \varepsilon\right] \xrightarrow{M\to\infty} 0. \tag{B.16}$$

■

Proof of Theorem A.2 (Item 2), Equation B.9. By assumption, we have $\operatorname{supp}\hat{p} \subset \operatorname{supp} Y_M$ for each M. Dropping the index M for brevity, we thus have

$$\mathbb{E}\left[\frac{\hat{p}(Y)}{p_Y(Y)}\right] = \int_{\operatorname{supp} Y} \hat{p}(y)\,\mathrm{d}y = \|\hat{p}\|. \tag{B.17}$$

Thus, we deduce from the law of total variance that

$$\begin{aligned} \operatorname{Var}\big[\hat{p}(Y)W_Y\big] &= \mathbb{E}\big[\operatorname{Var}\left[\hat{p}(Y)W_Y \mid Y\right]\big] + \operatorname{Var}\big[\mathbb{E}\left[\hat{p}(Y)W_Y \mid Y\right]\big] \\ &\geq \operatorname{Var}\big[\mathbb{E}\left[\hat{p}(Y)W_Y \mid Y\right]\big] = \operatorname{Var}\left[\frac{\hat{p}(Y)}{p_Y(Y)}\right] \\ &= \mathbb{E}\left[\left|\frac{\hat{p}(Y)}{p_Y(Y)} - \|\hat{p}\|\right|^2\right]. \end{aligned} \tag{B.18}$$

Since $\operatorname{Var}[\hat{p}(Y)W_Y]$ by assumption tends to 0, we reach convergence of p_Y to $\bar{p}$ in mean square:

$$\begin{aligned} \mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right|^2\right] &= \frac{1}{\|\hat{p}\|^2}\,\mathbb{E}\left[\left|\frac{\hat{p}(Y)}{p_Y(Y)} - \|\hat{p}\|\right|^2\right] \\ &\leq \frac{1}{\|\hat{p}\|^2}\operatorname{Var}\big[\hat{p}(Y)W_Y\big] \xrightarrow{M\to\infty} 0. \end{aligned} \tag{B.19}$$

■

Proof of Theorem A.2 (Item 3), Equation B.10. By the Cauchy-Schwarz inequality, convergence of a random variable in mean-square implies convergence in mean:

$$\mathbb{E}\left[|Z_i - Z_\infty|\right] \leq \sqrt{\mathbb{E}\left[1^2\right]}\sqrt{\mathbb{E}\left[|Z_i - Z_\infty|^2\right]} \xrightarrow{i\to\infty} 0. \tag{B.20}$$

By Equation B.9, $\mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right|\right] \xrightarrow{M\to\infty} 0$. Since we have $\operatorname{supp}\hat{p} \subset \operatorname{supp} Y$, we have $p_Y(y) - \hat{p}(y) = 0 - 0$ outside of $\operatorname{supp} Y$ for all M, and thus

$$\begin{aligned}\int_\Omega |p_Y(y) - \bar{p}(y)|\,\mathrm{d}y &= \int_{\operatorname{supp} Y} |p_Y(y) - \bar{p}(y)|\,\mathrm{d}y \\ &= \mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)} - 1\right|\right] \xrightarrow{M\to\infty} 0.\end{aligned} \tag{B.21}$$

■

Proof of Theorem A.2 (Item 4). Let G be the set of $y \in \Omega$ for which $p_{Y_M}(y)$ converges, and let $g(y)$ be the limit, i.e.,

$$g(y) = \lim_{M\to\infty} p_{Y_M}(y) \quad \text{for all } y \in G. \tag{B.22}$$

All subsequences $p_{Y_{a_k}}(y)$ also converge pointwise to $g(y)$ in G. By Equation B.10, $p_Y(y)$ converges to $\bar{p}$ in the Lebesgue L^1 sense:

$$\|\bar{p}(y) - p_{Y_M}(y)\|_{L^1} = \int_\Omega |\bar{p}(y) - p_{Y_M}(y)|\,\mathrm{d}y \xrightarrow{M\to\infty} 0. \tag{B.23}$$

Since p_{Y_M} converges to $\bar{p}$ in the L^1-norm, it converges also in the L^1-measure [189, p. 69]. Thus, it has a subsequence $p_{Y_{a_k}}$ that converges to $\bar{p}$ almost everywhere [189, p. 69]. But if $y \in G$, then $p_{Y_{a_k}}(y)$ also converges to $g(y)$, so we must have $g(y) = \bar{p}(y)$ almost everywhere in G. ■

Proof of Theorem A.2 (Item 5). Let X be distributed with PDF $\bar{p}$ and A be an arbitrary measurable subset of Ω. Then, by Theorem A.2 (Item 3),

$$\begin{aligned}\left|\Pr[Y \in A] - \Pr[X \in A]\right| &= \left|\int_A p_Y(y)\,\mathrm{d}y - \int_A \bar{p}(y)\,\mathrm{d}y\right| \\ &\leq \int_\Omega |p_Y(y) - \bar{p}(y)|\,\mathrm{d}y \xrightarrow{M\to\infty} 0.\end{aligned}$$

■

□

B.1.3 Asymptotic Variance, Section 6.4.2

Theorem A.3 (Asymptotic Variance). *In addition to the assumptions of Theorem A.2, assume that $f \geq 0$ and*

$$f \leq C_f\,\hat{p} \qquad \text{for some } C_f > 0. \tag{B.24}$$

Then, the generated samples Y cover the support of $\hat{p}$ and f, and

1. *$f(Y)W_Y$ approaches $f(Y)/\bar{p}(Y)$ in mean, mean square and probability, i.e.,*

$$\mathbb{E}\left[\left|f(Y)W_Y - \frac{f(Y)}{\bar{p}(Y)}\right|^p\right] \xrightarrow{M\to\infty} 0 \quad \text{for } p = 1, 2, \quad \text{and} \tag{B.25}$$

$$\Pr\left[\left|f(Y)W_Y - \frac{f(Y)}{\bar{p}(Y)}\right| > \varepsilon\right] \xrightarrow{M\to\infty} 0 \quad \text{for all } \varepsilon > 0. \tag{B.26}$$

2. *$f(Y)W_Y$ has asymptotically the variance of $f(X)/\bar{p}(X)$ where X has density $\bar{p}(X)$,*

$$\operatorname{Var}\left[f(Y)W_Y\right] \xrightarrow{M\to\infty} \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right]. \tag{B.27}$$

3. *If $\hat{p}(x) \propto f(x)$, then also*

$$\operatorname{Var}\left[f(Y)W_Y\right] \xrightarrow{M\to\infty} 0. \tag{B.28}$$

Proof. We first prove the support and then Item 1 – Item 3:

Proof of Theorem A.3, support. By assumption (Equation B.6), $\operatorname{supp}\hat{p} \subset \operatorname{supp} Y_M$ for all M. We also assume $f \leq C_f\hat{p}$ for some $C_f > 0$. Thus, $f(x) > 0$ implies $\hat{p}(x) > 0$, and we have $\operatorname{supp} f \subset \operatorname{supp}\hat{p} \subset \operatorname{supp} Y_M$. ∎

Proof of Theorem A.3 (Item 1), Equation B.25 and Equation B.26. We first prove convergence in mean square:

$$\begin{aligned}
&\mathbb{E}\left[\left|f(Y)W_Y - \frac{f(Y)}{\bar{p}(Y)}\right|^2\right] = \mathbb{E}\left[\left|\frac{f(Y)}{\hat{p}(Y)}\sum_{i=1}^{M} w_{M,i} - \frac{f(Y)}{\bar{p}(Y)}\right|^2\right] \\
&= \mathbb{E}\left[\frac{f(Y)^2}{\hat{p}(Y)^2}\left|\sum_{i=1}^{M} w_{M,i} - \frac{\hat{p}(Y)}{\bar{p}(Y)}\right|^2\right] = \mathbb{E}\left[\frac{f(Y)^2}{\hat{p}(Y)^2}\left|\sum_{i=1}^{M} w_{M,i} - \|\hat{p}\|\right|^2\right] \\
&\leq C_f^2\,\mathbb{E}\left[\left|\sum_{i=1}^{M} w_{M,i} - \|\hat{p}\|\right|^2\right] = C_f^2\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i}\right] \xrightarrow{M\to\infty} 0.
\end{aligned}$$

Convergence in mean square implies convergence in mean and in probability. E.g., by Chebyshev's inequality, given $\varepsilon > 0$, we have

$$\Pr\left[\left|f(Y)W_Y - \frac{f(Y)}{\bar{p}(Y)}\right| > \varepsilon\right] \leq \frac{1}{\varepsilon^2}\,\mathbb{E}\left[\left|f(Y)W_Y - \frac{f(Y)}{\bar{p}(Y)}\right|^2\right] \xrightarrow{M\to\infty} 0.$$

■

Proof of Theorem A.3 (Item 2), Equation B.27. We proceed in three steps.

- **Step 1.** We show that

$$\operatorname{Var}\left[f(Y)W_Y\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] \xrightarrow{M\to\infty} 0. \tag{B.29}$$

From the law of total variance we get

$$\operatorname{Var}\big[f(Y)W_Y\big] = \operatorname{Var}\big[\mathbb{E}\left[f(Y)W_Y|Y\right]\big] + \mathbb{E}\big[\operatorname{Var}\left[f(Y)W_Y \mid Y\right]\big],$$

which we rewrite as

$$\begin{aligned}\mathbb{E}\big[\operatorname{Var}\left[f(Y)W_Y \mid Y\right]\big] &= \operatorname{Var}\big[f(Y)W_Y\big] - \operatorname{Var}\big[\mathbb{E}\left[f(Y)W_Y|Y\right]\big]\\ &= \operatorname{Var}\big[f(Y)W_Y\big] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right].\end{aligned}$$

Since a conditional variance is non-negative, and we have

$$\begin{aligned}0 &\leq \operatorname{Var}\big[f(Y)W_Y\big] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] = \mathbb{E}\big[\operatorname{Var}\left[f(Y)W_Y \mid Y\right]\big]\\ &= \mathbb{E}\left[\operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\sum_{i=1}^{M} w_{M,i} \mid Y\right]\right] = \mathbb{E}\left[\frac{f(Y)^2}{\hat{p}(Y)^2}\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i} \mid Y\right]\right]\\ &\leq C_f^2\,\mathbb{E}\left[\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i} \mid Y\right]\right]\\ &\leq C_f^2\left(\mathbb{E}\left[\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i} \mid Y\right]\right] + \operatorname{Var}\left[\mathbb{E}\left[\sum_{i=1}^{M} w_{M,i} \mid Y\right]\right]\right)\\ &= C_f^2\operatorname{Var}\left[\sum_{i=1}^{M} w_{M,i}\right] \xrightarrow{M\to\infty} 0.\end{aligned} \tag{B.30}$$

- **Step 2.** We show that

$$\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] \xrightarrow{M\to\infty} 0 \tag{B.31}$$

where X has density $\bar{p}(x)$. We start by writing

$$\begin{aligned}
&\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] \\
&= \mathbb{E}\left[\frac{f(X)^2}{\bar{p}(X)^2}\right] - \mathbb{E}\left[\frac{f(X)}{\bar{p}(X)}\right]^2 - \mathbb{E}\left[\frac{f(Y)^2}{p_Y(Y)^2}\right] + \mathbb{E}\left[\frac{f(Y)}{p_Y(Y)}\right]^2 && \text{(B.32)} \\
&= \mathbb{E}\left[\frac{f(X)^2}{\bar{p}(X)^2}\right] - \mathbb{E}\left[\frac{f(Y)^2}{p_Y(Y)^2}\right]. && \text{(B.33)}
\end{aligned}$$

Since $\operatorname{supp} Y \subset \operatorname{supp} \hat{p}$ (Equation 6.15) and we assume $\operatorname{supp} \hat{p} \subset \operatorname{supp} Y$, we have $\operatorname{supp} \hat{p} = \operatorname{supp} Y$. With $\operatorname{supp} \bar{p} = \operatorname{supp} \hat{p}$, we continue the above as

$$\begin{aligned}
&= \int_{\operatorname{supp} Y} \bar{p}(x)\frac{f(x)^2}{\bar{p}(x)^2}\,\mathrm{d}x - \int_{\operatorname{supp} Y} p_Y(y)\frac{f(y)^2}{p_Y(y)^2}\,\mathrm{d}y \\
&= \int_{\operatorname{supp} Y} \bar{p}(y)\frac{f(y)^2}{\bar{p}(y)^2} - p_Y(y)\frac{f(y)^2}{p_Y(y)^2}\,\mathrm{d}y \\
&= \int_{\operatorname{supp} Y} p_Y(y)\left(\frac{f(y)^2}{\bar{p}(y)^2}\right)\left(\frac{\bar{p}(y)}{p_Y(y)} - \frac{\bar{p}(y)^2}{p_Y(y)^2}\right)\mathrm{d}y \\
&= \mathbb{E}\left[\frac{f(Y)^2}{\bar{p}(Y)^2}\left(\frac{\bar{p}(Y)}{p_Y(Y)} - \frac{\bar{p}(Y)^2}{p_Y(Y)^2}\right)\right],
\end{aligned}$$

and thus,

$$\begin{aligned}
&\left|\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right]-\operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right]\right| \\
&\leq \mathbb{E}\left[\frac{f(Y)^2}{\bar{p}(Y)^2}\left|\frac{\bar{p}(Y)}{p_Y(Y)}-\frac{\bar{p}(Y)^2}{p_Y(Y)^2}\right|\right] \\
&\leq \|\hat{p}\|^2 C_f^2\, \mathbb{E}\left[\left|\frac{\bar{p}(Y)}{p_Y(Y)}-\frac{\bar{p}(Y)^2}{p_Y(Y)^2}\right|\right] \\
&= \|\hat{p}\|^2 C_f^2\, \mathbb{E}\left[\left|1-\left(1-\frac{\bar{p}(Y)}{p_Y(Y)}\right)\right|\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|\right] \\
&\leq \|\hat{p}\|^2 C_f^2\, \mathbb{E}\left[\left(1+\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|\right)\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|\right] \\
&= \|\hat{p}\|^2 C_f^2\left(\mathbb{E}\left[\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|\right]+\mathbb{E}\left[\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right]\right) \\
&\leq \|\hat{p}\|^2 C_f^2\left(\sqrt{\mathbb{E}\left[\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right]}+\mathbb{E}\left[\left|1-\frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right]\right) \qquad \text{(B.34)}\\
&\xrightarrow{M\to\infty} 0
\end{aligned}$$

by Theorem A.2 (Item 2).

- **Step 3.** Combining steps 1 and 2, we reach

$$\begin{aligned}
&\operatorname{Var}\left[f(Y)W_Y\right]-\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] \qquad \text{(B.35)}\\
=&\left(\operatorname{Var}\left[f(Y)W_Y\right]-\operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right]\right)-\left(\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right]-\operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right]\right) \\
&\xrightarrow{M\to\infty} 0.
\end{aligned}$$

■

Proof of Theorem A.3 (Item 3). Substituting $\hat{p}(x) = Cf(x)$, i.e., $\bar{p}(x) = f(x)/\|f\|$, to the previous result, yields

$$\operatorname{Var}\left[f(Y)W_Y\right]-\operatorname{Var}\left[\frac{f(X)\|f\|}{f(X)}\right] \xrightarrow{M\to\infty} 0,$$

i.e.,

$$\operatorname{Var}\left[f(Y)W_Y\right] \xrightarrow{M\to\infty} 0.$$

■

□

B.1.4 Resampling Weight Bounds, Section 6.4.6

Theorem A.4 (Resampling Weight Bounds)**.** *Let the resampling weights w_i of input samples X_i be given by Equation 6.19, and associate all source domains Ω_i with target distributions $\hat{p}_i$. Let R be the indices of the canonical samples, and assume $|R| \geq 1$.*

If m_i are given by one of the MIS weight schemes defined in Equations 6.36, 6.37 or 6.38, and sample X_i is reasonably distributed for integrating $\hat{p}_i$ (i.e., $\hat{p}_i(X_i)W_i \leq C_i$ for some C_i), then the resampling weight of X_i is bounded as

$$w_i \leq \frac{C_i}{|R|}. \tag{B.36}$$

Proof. Section B.4.3. □

B.1.5 Proof of Theorem 1, Section 6.4.3

Proof. We continue the proof of Theorem A.3 (Item 2), and deduce

$$\begin{aligned}
&\operatorname{Var}\left[f(Y)W_Y\right] - \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] \\
&= \operatorname{Var}\left[f(Y)W_Y\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] + \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] - \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] \\
&\leq \operatorname{Var}\left[f(Y)W_Y\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] + \left|\operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] - \operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right]\right|.
\end{aligned}$$

We denote $V = \operatorname{Var}\left[\sum_{i=1}^{M} w_i\right]$ and derive from Equation B.18 in the proof of Theorem A.2 that

$$\mathbb{E}\left[\left|1 - \frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right] \leq \frac{\operatorname{Var}\left[\hat{p}(Y)W_Y\right]}{\|\hat{p}\|^2} = \frac{V}{\|\hat{p}\|^2}. \tag{B.37}$$

We combine this with Equation B.34 to get

$$\left|\operatorname{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right]\right| \tag{B.38}$$

$$\leq \|\hat{p}\|^2 C_f^2 \left(\sqrt{\mathbb{E}\left[\left|1 - \frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right]} + \mathbb{E}\left[\left|1 - \frac{\bar{p}(Y)}{p_Y(Y)}\right|^2\right]\right) \tag{B.39}$$

$$\leq \|\hat{p}\|^2 C_f^2 \left(\sqrt{\frac{V}{\|\hat{p}\|^2}} + \frac{V}{\|\hat{p}\|^2}\right) = C_f^2\left(\|\hat{p}\|\sqrt{V} + V\right). \tag{B.40}$$

We then borrow from Equation B.30 that

$$0 \leq \operatorname{Var}\left[f(Y)W_Y\right] - \operatorname{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] \leq C_f^2 \operatorname{Var}\left[\sum_{i=1}^{M} w_i\right] = C_f^2\, V,$$

and reach

$$\begin{aligned}
&\mathrm{Var}\left[f(Y)W_Y\right] - \mathrm{Var}\left[\frac{f(X)}{\bar{p}(X)}\right] \\
&\leq \mathrm{Var}\left[f(Y)W_Y\right] - \mathrm{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] + \left|\mathrm{Var}\left[\frac{f(Y)}{p_Y(Y)}\right] - \mathrm{Var}\left[\frac{f(X)}{\bar{p}(X)}\right]\right|. \\
&\leq C_f^2 V + C_f^2\left(\|\hat{p}\|\sqrt{V} + V\right) = C_f^2\sqrt{V}\left(\|\hat{p}\| + 2\sqrt{V}\right). \qquad \text{(B.41)}
\end{aligned}$$

□

B.2 Convergence With Dependent Samples

In Section 6.4.7 we assume dependent input samples such that

1. the ratio of canonical samples, $|R|/M$, never falls below a positive constant γ for large enough M,
2. there exists a $C > 0$ such that $w_i \leq C/|R|$ for all i,
3. there exists a non-negative sequence b_k such that the correlation $\rho_{i,i+k} \leq b_k$ for all i, and $b_k \to 0$.

Then,

$$\mathrm{Var}\left[\sum_{i=1}^{M} w_i\right] = \sum_{i=1}^{M}\mathrm{Var}\left[w_i\right] + 2\sum_{i=1}^{M}\sum_{k=1}^{M-i}\mathrm{Cov}(w_i, w_{i+k}) \qquad \text{(B.42)}$$

converges to zero:

The convergence of the first term is proved in Section 6.4.7, and for the second term we have

$$\begin{aligned}
\sum_{i=1}^{M}\sum_{k=1}^{M-i}\mathrm{Cov}(w_i, w_{i+k}) &= \sum_{i=1}^{M}\sum_{k=1}^{M-i}\rho_{i,i+k}\sqrt{\mathrm{Var}\, w_i\ \mathrm{Var}\, w_{i+k}} && \text{(B.43)}\\
&\leq \sum_{i=1}^{M}\sum_{k=1}^{M-i}\max(0, \rho_{i,i+k})\frac{C^2}{4|R|^2} \leq \sum_{i=1}^{M}\sum_{k=1}^{M-i} b_k\frac{C^2}{4M^2\gamma^2}, && \text{(B.44)}\\
&= \frac{C^2}{4\gamma^2}\frac{1}{M^2}\sum_{k=1}^{M}\sum_{i=1}^{M-k} b_k = \frac{C^2}{4\gamma^2}\frac{1}{M^2}\sum_{k=1}^{M}(M-k)b_k && \text{(B.45)}\\
&\leq \frac{C^2}{4\gamma^2}\left(\frac{1}{M}\sum_{k=1}^{M} b_k\right) \xrightarrow{M\to\infty} 0. && \text{(B.46)}
\end{aligned}$$

To reach Equation B.44, we used Popoviciu's inequality: since $0 \leq w_i \leq C/|R|$, we know $\mathrm{Var}\, w_i \leq \frac{C^2}{4|R|^2}$. The next step used $|R|/M \geq \gamma$, and for Equation B.45 we reversed the summation order: $\sum_{i=1}^{M}\sum_{k=1}^{M-i} = \sum_{k=1}^{M}\sum_{i=1}^{M-k}$. The mean of b_k converges to zero since b_k converges to zero, and Equation B.46 implies

$$\operatorname{Var}\left[\sum_{i=1}^{M} w_i\right] \xrightarrow{M\to\infty} 0.$$

We can slightly generalize the result: assume that

$$|R| \geq c_M M^{0.5}\sqrt{\sum_{i=1}^{M} b_i}\,, \tag{B.47}$$

where (c_M) is any non-negative sequence that approaches infinity. Then, like above,

$$\sum_{i=1}^{M}\sum_{k=1}^{M-i}\operatorname{Cov}(w_i, w_{i+k}) \leq \sum_{i=1}^{M}\sum_{k=1}^{M-i}\max(0,\rho_{i,i+k})\frac{C^2}{4\,|R|^2} \tag{B.48}$$

$$= \sum_{k=1}^{M}\sum_{i=1}^{M-k}\max(0,\rho_{i,i+k})\frac{C^2}{4\,|R|^2} \leq \sum_{k=1}^{M}(M-k)b_k\frac{C^2}{4\,|R|^2} \tag{B.49}$$

$$\leq \left(\sum_{k=1}^{M} b_k\right)\frac{MC^2}{4\,|R|^2} \leq \left(\sum_{k=1}^{M} b_k\right)\frac{MC^2}{4c_M^2 M\left(\sum_{i=1}^{M} b_i\right)} \tag{B.50}$$

$$= \frac{C^2}{4c_M^2} \xrightarrow{M\to\infty} 0. \tag{B.51}$$

B.3 Mathematical Notes

B.3.1 Constraints on w_i for Zero Bias in Section 6.3.3

We assume, according to the section, that

$$g_i(x) = [x \in \mathcal{D}(T_i)]\; c_i(y_i)\cdot f(y_i)\left|\frac{\partial T_i}{\partial x}\right|,$$

where y_i is a shorthand for $T_i(x)$, and $[\cdot]$ is 1 if $\cdot$ is true and 0 otherwise, and that either

- (easy case) $w_i > 0$ exactly when $X_i \in \mathcal{D}(T_i)$ and $Y_i = T_i(X_i) \in \operatorname{supp}\hat{p}$, otherwise $w_i = 0$, or

- (general case) w_i are also allowed to be 0 also when $c_i(Y_i) = 0$ or $W_i = 0$, and Equation 6.17 holds for all $y \in \operatorname{supp}\hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$.

Then, we show that the derived estimator is unbiased,

$$E\left[g_s(X_s)\frac{\sum_{j=1}^{M} w_j}{w_s} f(X_s) W_s\right] = \int_{\operatorname{supp} Y} f(y)\,\mathrm{d}y, \tag{B.52}$$

and that Equation 6.15 holds, i.e.,

$$\operatorname{supp} Y = \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i). \tag{B.53}$$

Let us first prove Equation B.53 in both cases.

Proof. Easy case. If $y \in \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$, then $y = T_i(x_i)$ for some $x_i \in \operatorname{supp} X_i$ where x_i can be sampled with a positive PDF. Since also $y \in \operatorname{supp} \hat{p}$, we have $\hat{p}(y) = \hat{p}(T_i(x_i)) > 0$, and therefore by assumption $w_i > 0$. Thus, we have an x_i with positive PDF, $w_i > 0$, and $y = T_i(x_i)$, and we have a way of sampling y with a positive PDF, i.e., $y \in \operatorname{supp} Y$. We showed that $\operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i) \subset \operatorname{supp} Y$.

If, on the other hand, $y \in \operatorname{supp} Y$, the PDF of sampling y is positive. Thus, there exists an x_i with a positive PDF and a positive selection probability for $y = T_i(x_i)$. Hence $w_i > 0$, which implies $\hat{p}(y) > 0$, and hence $y \in \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$. Combined with the previous, we have $\operatorname{supp} Y = \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$.

General case. Let $y \in \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$. Let I be the set of indices for which $y \in T_i(\operatorname{supp} X_i)$. Since $\sum_{i \in I} c_i(y) = 1$ by assumption, there exists at least one i such that $c_i(y) > 0$, and therefore an x_i such that $y = T_i(x_i)$ with $p_{X_i}(x_i) > 0$. Since $0 < p_{X_i}(x_i) = 1/\,\mathbb{E}\left[W_i \mid X_i = x_i\right]$, the conditional expectation of W_i is positive and hence we have $\Pr\left[W_i > 0 \mid X_i = x_i\right] > 0$ and $p_Y(y) > 0$, i.e., $y \in \operatorname{supp} Y$.

Assume then that $y \in \operatorname{supp} Y$. The PDF of generating y is positive, so there have to exist i and $x_i \in \Omega_i$ such that $p_{X_i}(x_i) > 0$, $y = T_i(x_i)$, and with a positive conditional probability $w_i > 0$. If w_i may be positive, it follows that $y \in \operatorname{supp} \hat{p}$, and hence $y \in \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$. Combined with the previous, we have $\operatorname{supp} Y = \operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)$. □

Next, we will prove Equation B.52.

Proof. First,

$$\mathbb{E}\left[g_s(X_s)\frac{\sum_{j=1}^M w_j}{w_s}W_s\right] \tag{B.54}$$

$$= \mathbb{E}\left[\sum_{s=1}^M [w_s > 0]\frac{w_s}{\sum_{j=1}^M w_j}g_s(X_s)\frac{\sum_{j=1}^M w_j}{w_s}W_s\right] \tag{B.55}$$

$$= \mathbb{E}\left[\sum_{s=1}^M [w_s > 0]g_s(X_s)W_s\right]. \tag{B.56}$$

Next, we substitute the definition of g_s and reach

$$= \mathbb{E}\left[\sum_{s=1}^M [w_s > 0][X_s \in \mathcal{D}(T_s)]\; c_s(Y_s)f(Y_s)\left|\frac{\partial T_s}{\partial X_s}\right| W_s\right]. \tag{B.57}$$

We add in the assumption which holds in both cases, $w_s = 0$ if $\hat{p}(Y_s) = 0$, and reach

$$= \mathbb{E}\left[\sum_{s=1}^M [Y_s \in \operatorname{supp}\hat{p}][w_s > 0][X_s \in \mathcal{D}(T_s)]\; c_s(Y_s)f(Y_s)\left|\frac{\partial T_s}{\partial X_s}\right| W_s\right]. \tag{B.58}$$

Next we substitute $[w_s > 0] = 1 - [w_s = 0]$, and reach

$$\begin{aligned} &= \mathbb{E}\left[\sum_{s=1}^M [Y_s \in \operatorname{supp}\hat{p}][X_s \in \mathcal{D}(T_s)]\; c_s(Y_s)f(Y_s)\left|\frac{\partial T_s}{\partial X_s}\right| W_s\right] \\ &\quad - \mathbb{E}\left[\sum_{s=1}^M [Y_s \in \operatorname{supp}\hat{p}][w_s = 0][X_s \in \mathcal{D}(T_s)]\; c_s(Y_s)f(Y_s)\left|\frac{\partial T_s}{\partial X_s}\right| W_s\right]. \end{aligned} \tag{B.59}$$

Using the definition of unbiased contribution weights (everything except W_s is a function of X_s), we get for the first term,

$$\mathbb{E}\left[\sum_{s=1}^M [Y_s \in \operatorname{supp}\hat{p}][X_s \in \mathcal{D}(T_s)]\; c_s(Y_s)f(Y_s)\left|\frac{\partial T_s}{\partial X_s}\right| W_s\right] \tag{B.60}$$

$$= \sum_{s=1}^M \int_{\operatorname{supp} X_s} [y_s \in \operatorname{supp}\hat{p}][x_s \in \mathcal{D}(T_s)]\; c_s(y_s)f(y_s)\left|\frac{\partial T_s}{\partial x_s}\right| \mathrm{d}x_s \tag{B.61}$$

$$= \sum_{s=1}^M \int_{\mathcal{D}(T_s)} [y_s \in \operatorname{supp}\hat{p}][x_s \in \operatorname{supp} X_s]\; c_s(y_s)f(y_s)\left|\frac{\partial T_s}{\partial x_s}\right| \mathrm{d}x_s, \tag{B.62}$$

which, with a change of variables $y = T_s(x_s)$ for each of the terms, and then denoting $x_s = T_s^{-1}(y)$, simplifies into

$$= \sum_{s=1}^M \int_{\mathcal{I}(T_s)} [y \in \operatorname{supp}\hat{p}][x_s \in \operatorname{supp} X_s]\; c_s(y)f(y)\,\mathrm{d}y \tag{B.63}$$

$$= \sum_{s=1}^M \int_{\operatorname{supp}\hat{p}} [y \in \mathcal{I}(T_s)][x_s \in \operatorname{supp} X_s]\; c_s(y)f(y)\,\mathrm{d}y \tag{B.64}$$

$$= \int_{\operatorname{supp}\hat{p}} \left(\sum_{s=1}^M [y \in \mathcal{I}(T_s)][x \in \operatorname{supp} X_s]\; c_s(y)\right) f(y)\,\mathrm{d}y. \tag{B.65}$$

We then write the product of the brackets as a summation condition and reach

$$= \int_{\operatorname{supp} \hat{p}} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} c_s(y_s) \right) f(y) \, \mathrm{d}y \tag{B.66}$$

$$= \int_{\operatorname{supp} \hat{p} \cap \bigcup_i T_i(\operatorname{supp} X_i)} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} c_s(y_s) \right) f(y) \, \mathrm{d}y \tag{B.67}$$

$$= \int_{\operatorname{supp} Y} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} c_s(y) \right) f(y) \, \mathrm{d}y \tag{B.68}$$

$$= \int_{\operatorname{supp} Y} f(y) \, \mathrm{d}y. \tag{B.69}$$

$$\tag{B.70}$$

For the method be unbiased for integrating f over $\operatorname{supp} Y$, the second term,

$$\mathbb{E}\left[\sum_{s=1}^{M} [Y_s \in \operatorname{supp} \hat{p}][w_s = 0][X_s \in \mathcal{D}(T_s)] \, c_s(Y_s) f(Y_s) \left| \frac{\partial T_s}{\partial X_s} \right| W_s \right],$$

must be zero.

For the easier case, $w_i > 0$ if and only if $X_i \in \mathcal{D}(T_i)$ and $Y_i = T_i(X_i) \in \operatorname{supp} \hat{p}$. The above second term contains for each s the factor

$$[Y_s \in \operatorname{supp} \hat{p}][X_s \in \mathcal{D}(T_s)][w_s = 0] \tag{B.71}$$

$$= [w_s > 0][w_s = 0] = 0, \tag{B.72}$$

and thus the second term is zero and the estimator is unbiased.

For the general case, w_i is additionally allowed to be 0 when either $W_i = 0$ or $c_i(Y_i) = 0$. The second term is also then zero: for it to be non-zero, we need to have $w_s = 0$ for some s. However, if $w_s = 0$, then by assumption, either $X_s \notin \mathcal{D}(T_s)$, $Y_s \notin \operatorname{supp} \hat{p}$, $c_s(Y_s) = 0$, or $W_s = 0$. It is easy to check all these cases: in all cases the second term is zero. The estimator is unbiased. □

B.3.2 Non-Negativity of m_i and W_i in Section 6.3.4

Here we prove that in Equation 6.19, we must require $m_i \geq 0$ and $W_i \geq 0$ to guarantee non-negative probabilities.

Proof. The selection probabilities $\Pr[s = i] = w_i / \sum_{j=1}^{M} w_j$ all need to be non-negative. They are divided by the same denominator, which flips the sign of either none or all the $w_i / \sum_{j=1}^{M} w_j$ expressions. Therefore, all w_i must have the same sign, so that the probabilities can all be non-negative. If we had $w_i \leq 0$ for all i, we could simply flip the signs of the w_i to reach $w_i \geq 0$; without loss of generality, we restrict ourselves to the case $w_i \geq 0$.

If it is the case that $w_i \geq 0$, then it is also the case that

$$\mathbb{E}[w_i \mid X_i] \geq 0 \tag{B.73}$$

almost surely.

Next, we substitute Equation 6.19 for w_i. If $x \notin \mathcal{D}(T_i)$, we have $w_i = 0$. Otherwise, denoting $y = T_i(x)$ and $Y_i = T_i(X_i)$, we must have

$$0 \leq \mathbb{E}[w_i \mid X_i = x] = \mathbb{E}\left[m_i(Y_i)\hat{p}(Y_i)W_i \left|\frac{\partial T_i}{\partial X_i}\right| \mid X_i = x\right] \tag{B.74}$$

$$= m_i(y)\hat{p}(y)\left|\frac{\partial T_i}{\partial x}\right| \mathbb{E}[W_i \mid X_i = x] \tag{B.75}$$

$$= m_i(y)\hat{p}(y)\left|\frac{\partial T_i}{\partial x}\right| \frac{1}{p_{X_i}(x)}. \tag{B.76}$$

Since $\hat{p}(y)$, $\left|\frac{\partial T_i}{\partial x}\right|$ and $p_{X_i}(x)$ are all non-negative and the full product is non-negative, $m_i(y)$ must also be non-negative.

Looking at Equation 6.19, since we have $m_i \geq 0$, $\hat{p}(y) \geq 0$ and $\left|\frac{\partial T_i}{\partial x}\right| \geq 0$, and their product with W_i is $w_i \geq 0$, we must also have $W_i \geq 0$. □

B.3.3 Resampling MIS Must Be Positive in Support of c_i in Section 6.3.4

To guarantee unbiased integration, the resampling MIS weights m_i must fulfill $m_i(y) > 0$ whenever $c_i(y) \neq 0$ in addition to non-negativity and Equation 6.20:

Proof. Substituting g_i into the left-hand-side of Equation 6.10 yields

$$\mathbb{E}\left[g_s(X_s)\frac{\sum_{j=1}^{M} w_j}{w_s}W_s\right] = \mathbb{E}\left[[w_s > 0]g_s(X_s)\frac{\sum_{j=1}^{M} w_j}{w_s}W_s\right] \tag{B.77}$$

$$= \mathbb{E}\left[\sum_{s=1}^{M}[w_s > 0]\frac{w_s}{\sum_{j=1}^{M} w_j}g_s(X_s)\frac{\sum_{j=1}^{M} w_j}{w_s}W_s\right] \tag{B.78}$$

$$= \sum_{s=1}^{M}\mathbb{E}\left[[w_s > 0]g_s(X_s)W_s\right]. \tag{B.79}$$

We know that w_s (Equation 6.19) is positive if and only if $X_s \in \mathcal{D}(T_s)$, $m_s(Y_s) > 0$, $\hat{p}(Y_s) > 0$, $W_s > 0$ and $\left|\frac{\partial T_s}{\partial X_s}\right| > 0$. We assume $W_s \geq 0$, and the case $W_s = 0$ is already handled by the factor W_s. The Jacobian determinant is nonzero with probability 1 in $\mathcal{D}(T_i)$ since T_i is bijective. Hence, substituting g_s, we reach

$$= \sum_{s=1}^{M} \mathbb{E}\left[[X_s \in \mathcal{D}(T_s)][m_s(Y_s), \hat{p}(Y_s) > 0]\; c_s(Y_s) f(Y_s) \left|\frac{\partial T_s}{\partial X_s}\right| W_s\right].$$

Everything left from the unbiased contribution weight W_s is a function of X_s. Hence, by the definition of unbiased contribution weights, we reach

$$= \sum_{s=1}^{M} \int_{\operatorname{supp} X_s} [x_s \in \mathcal{D}(T_s)][m_s(y_s), \hat{p}(y_s) > 0]\; c_s(y_s) f(y_s) \left|\frac{\partial T_s}{\partial x_s}\right| \mathrm{d}x_s.$$

Swapping the integration domain and $[x_s \in \mathcal{D}(T_s)]$, we reach

$$= \sum_{s=1}^{M} \int_{\mathcal{D}(T_s)} [x_s \in \operatorname{supp} X_s][m_s(y_s), \hat{p}(y_s) > 0]\; c_s(y_s) f(y_s) \left|\frac{\partial T_s}{\partial x_s}\right| \mathrm{d}x_s,$$

and the change of variables $y = T_s(x_s)$ yields

$$= \sum_{s=1}^{M} \int_{\mathcal{I}(T_s)} [x_s \in \operatorname{supp} X_s][m_s(y), \hat{p}(y) > 0]\; c_s(y) f(y)\, \mathrm{d}y.$$

Swapping the integration domain and $[\hat{p}(y) > 0]$ yields

$$= \sum_{s=1}^{M} \int_{\operatorname{supp} \hat{p}} [x_s \in \operatorname{supp} X_s][y \in \mathcal{I}(T_s)][m_s(y) > 0]\; c_s(y) f(y)\, \mathrm{d}y,$$

which allows moving the sum inside the integral:

$$= \int_{\operatorname{supp} \hat{p}} \left(\sum_{s=1}^{M} [x_s \in \operatorname{supp} X_s][y \in \mathcal{I}(T_s)][m_s(y) > 0]\; c_s(y)\right) f(y)\, \mathrm{d}y.$$

Next, we simplify the first two indicators into the summation condition:

$$= \int_{\operatorname{supp} \hat{p}} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} [m_s(y) > 0]\; c_s(y)\right) f(y)\, \mathrm{d}y. \tag{B.80}$$

The integrand is zero unless the sum contains at least one index and we can shrink the integration domain accordingly:

$$= \int_{\operatorname{supp} \hat{p} \cap \bigcup_{i=1}^{M} T_i(\operatorname{supp} X_i)} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} [m_s(y) > 0]\; c_s(y)\right) f(y)\, \mathrm{d}y.$$

By Equation 6.15, this domain is exactly $\operatorname{supp}(Y)$:

$$= \int_{\operatorname{supp} Y} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} [m_s(y) > 0]\, c_s(y) \right) f(y)\, \mathrm{d}y. \tag{B.81}$$

Finally, if we assume $m_s(y) > 0$ whenever $c_s(y) \neq 0$, by the constraint of the contribution MIS weights (Equation 6.17), we reach

$$= \int_{\operatorname{supp} Y} \left(\sum_{\substack{s=1 \\ y \in T_s(\operatorname{supp} X_s)}}^{M} c_s(y) \right) f(y)\, \mathrm{d}y \tag{B.82}$$

$$= \int_{\operatorname{supp} Y} f(y)\, \mathrm{d}y. \tag{B.83}$$

If we assumed a non-zero probability that $m_s(y) = 0$ while $c_s(y) \neq 0$, the multiplier in front of $f(y)$ in Equation B.81 would not simplify to 1 with the constraints of c_i, and the result would be wrong. $\square$

B.4 Derivation of the Resampling MIS Weights

In this section we first derive the generalized Talbot MIS (Equation 6.36) and Pairwise MIS (Equation 6.37, Equation 6.38) weights from the requirement that the resampling weights w_i given by Equation 6.19 must have a finite upper bound.

We then derive the upper bounds for the resampling weights w_i with the above MIS weights schemes, and for variants of the MIS weights that use tractable PDFs p_i instead of $\hat{p}_i$.

We assume that of the M input samples X_i, indices in the set R are canonical (Definition 6.4.2), i.e., their domain is Ω, the shift mapping is identity, and the target density $\hat{p}_i = \hat{p}$. The number of canonical samples is denoted $|R|$.

B.4.1 Generalizing Talbot MIS Weights

We first require that the resampling weights stay bounded, and derive MIS weights m_i that fulfill this condition. Denote $Y_i = T_i(X_i)$ and assume that $Y_i \in T_i(\operatorname{supp} X_i)$. The resampling weight of X_i is then, by Equation 6.19,

$$w_i = m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial Y_i}{\partial X_i} \right|. \tag{B.84}$$

Assuming that $\hat{p}_i(X_i)W_i \leq C_i$, we can bound the above as

$$w_i = m_i(Y_i) \cdot \frac{\hat{p}(Y_i)\hat{p}_i(X_i)W_i}{\hat{p}_i(X_i)} \cdot \left|\frac{\partial Y_i}{\partial X_i}\right| \tag{B.85}$$

$$\leq m_i(Y_i) \cdot \frac{\hat{p}(Y_i)\, C_i}{\hat{p}_i(X_i)} \cdot \left|\frac{\partial Y_i}{\partial X_i}\right|. \tag{B.86}$$

We require this to be at most some $\tilde{C}_i$, which we may choose freely as long as we still find suitable functions m_i. Then, $w_i \leq \tilde{C}_i$ will also hold:

$$w_i \leq m_i(Y_i) \cdot \frac{\hat{p}(Y_i)C_i}{\hat{p}_i(X_i)} \cdot \left|\frac{\partial Y_i}{\partial X_i}\right| \leq \tilde{C}_i. \tag{B.87}$$

The latter inequality is equivalent to

$$m_i(Y_i) \leq \frac{\tilde{C}_i}{C_i} \frac{\hat{p}_i(X_i)\left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}{\hat{p}(Y_i)}. \tag{B.88}$$

We observe that if j is any canonical index, then we have $\hat{p}(Y_i) = \hat{p}_j(T_j^{-1}(Y_i))\left|\frac{\partial T_j^{-1}}{\partial Y_i}\right|$, and the numerator and denominator begin to look similar:

$$m_i(Y_i) \leq \frac{\tilde{C}_i}{C_i} \frac{\hat{p}_i\left(T_i^{-1}(Y_i)\right)\left|\frac{\partial T_i^{-1}}{\partial Y_i}\right|}{\hat{p}_j(T_j^{-1}(Y_i))\left|\frac{\partial T_j^{-1}}{\partial Y_i}\right|} = \frac{\tilde{C}_i}{C_i} \frac{\hat{p}_{\leftarrow i}(Y_i)}{\hat{p}_{\leftarrow j}(Y_i)}. \tag{B.89}$$

Writing the expressions in terms of $\hat{p}_{\leftarrow i}$ and $\hat{p}_{\leftarrow j}$ is justified since $Y_i = T_i(X_i)$ was sampled (hence $Y_i \in T_i(\operatorname{supp} X_i)$), which implies $Y_i \in \operatorname{supp} \hat{p}$. Since j is a canonical index, $\operatorname{supp} \hat{p} \subset \operatorname{supp} X_j = \operatorname{supp} T_j(\operatorname{supp} X_j)$, and hence $Y_i \in \operatorname{supp} T_j(X_j)$.

If m_i is such that it fulfills the above inequality but with a larger denominator, it will also fulfill the above inequality. We make the denominator of m_i symmetric by summing over *all* indices $j \in \{1, \ldots, M\}$. We additionally choose $\tilde{C}_i = C_i$, leading to

$$m_i(y) = \frac{\hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} \hat{p}_{\leftarrow j}(y)}. \tag{B.90}$$

These m_i fulfill Equation 6.20 (see the definition of $\hat{p}_{\leftarrow i}$, Equation 6.35) and are valid, non-negative resampling MIS weights.

B.4.2 Generalizing Pairwise MIS Weights

In order to derive the generalized Pairwise MIS weights, we proceed as before until Equation B.88,

$$m_i(Y_i) \leq \frac{\tilde{C}_i}{C_i} \frac{\hat{p}_{\leftarrow i}(Y_i)}{\hat{p}(Y_i)}, \tag{B.91}$$

but then treat canonical samples $i \in R$ and non-canonical indices $i \notin R$ differently. Instead of including all the indices in the denominator like before, we only increase the denominator by the term corresponding to index i, with a positive multiplier α_i. We set for non-canonical samples

$$m_i(y) = \frac{\tilde{C}_i}{C_i} \frac{\hat{p}_{\leftarrow i}(y)}{\hat{p}(y) + \alpha_i\, \hat{p}_{\leftarrow i}(y)} \quad \text{for } i \notin R, \tag{B.92}$$

and observe that this choice fulfills Equation B.91. We still need m_i to sum to 1 over the i that can generate y, in order to fulfill Equation 6.20, so we simply divide the remainder uniformly to the canonical samples $i \in R$,

$$m_i(y) = \frac{1}{|R|}\left(1 - \sum_{j \notin R} m_j(y)\right) \quad \text{if i} \in \text{R}. \tag{B.93}$$

Different choices for $\tilde{C}_i$ and α_i yield a family of potential MIS weights: denoting $\beta_i = \tilde{C}_i / C_i$, we reach

$$\begin{aligned} m_i(y) &= \frac{1}{|R|}\left(1 - \sum_{j \notin R} \beta_j \frac{\hat{p}_{\leftarrow j}(y)}{\hat{p}(y) + \alpha_j \hat{p}_{\leftarrow j}(y)}\right) && \text{(B.94)} \\ &= \frac{1}{|R|}\left(1 - \sum_{j \notin R} \frac{\beta_j}{\alpha_j} + \sum_{j \notin R} \frac{\beta_j}{\alpha_j} \frac{\hat{p}(y)}{\hat{p}(y) + \alpha_j \hat{p}_{\leftarrow j}(y)}\right) \quad \text{if} \in R \\ m_i(y) &= \beta_i \frac{\hat{p}_{\leftarrow i}(y)}{\hat{p}(y) + \alpha_i \hat{p}_{\leftarrow i}(y)} \quad \text{if } i \notin R. && \text{(B.95)} \end{aligned}$$

Restricting this family by requiring that the parameters do not depend on i, and denoting $\alpha_i = \alpha$ and $\kappa = \sum_{i \notin R} \beta_i / \alpha_i = (M - |R|)\beta / \alpha$, we reach the family

$$m_i(y) = \frac{1}{|R|}\left(1 - \kappa + \frac{\kappa}{M - |R|} \sum_{j \notin R} \frac{\hat{p}(y)}{\hat{p}(y) + \alpha \hat{p}_{\leftarrow j}(y)}\right) \quad \text{if } i \in R \tag{B.96}$$

$$m_i(y) = \alpha \frac{\kappa}{M - |R|} \frac{\hat{p}_{\leftarrow i}(y)}{\hat{p}(y) + \alpha \hat{p}_{\leftarrow i}(y)} \quad \text{if } i \notin R. \tag{B.97}$$

Since MIS weights must be non-negative and sum to one, we must have $0 \leq m_i \leq 1$ for all i and y. We must generally have $\kappa \leq 1$ since otherwise $m_i(y)$ could be negative in Equation B.96. Since $\alpha > 0$ and we must have $m_i(y) \geq 0$ in Equation B.97, we must also have $0 \leq \kappa$. We have $0 \leq \kappa \leq 1$, and we interpret the above MIS weights as linear interpolation between uniformly choosing one of the canonical samples ($\kappa = 0$, $m_i = 1/|R|$ for $i \in R$, $m_i = 0$ for $i \notin R$), and a fundamental MIS scheme ($\kappa = 1$) parametrized by α:

$$m_i(y) = \frac{1}{|R|}\left(\frac{1}{M-|R|}\sum_{j\notin R}\frac{\hat{p}(y)}{\hat{p}(y)+\alpha\hat{p}_{\leftarrow j}(y)}\right) \quad \text{if } i \in R \tag{B.98}$$

$$m_i(y) = \alpha\frac{1}{M-|R|}\frac{\hat{p}_{\leftarrow i}(y)}{\hat{p}(y)+\alpha\hat{p}_{\leftarrow i}(y)} \quad \text{if } i \notin R. \tag{B.99}$$

• **The uniform case.** To find a sensible value for α for the fundamental MIS scheme, we consider the simple case when all X_i are i.i.d. with $\hat{p}_i = \hat{p}$ for all i. In this case, we have no reason to favor any of the samples and we should have $m_1 = \cdots = m_M = 1/M$, yielding

$$m_i(y) = \frac{1}{|R|}\left(\frac{1}{M-|R|}\sum_{j\notin R}\frac{1}{1+\alpha}\right) = \frac{1}{M} \quad \text{if } i \in R \tag{B.100}$$

$$m_i(y) = \alpha\frac{1}{M-|R|}\frac{1}{1+\alpha} = \frac{1}{M} \quad \text{if } i \notin R, \tag{B.101}$$

from which we solve

$$\alpha = \frac{M}{|R|} - 1. \tag{B.102}$$

In the fundamental case $\kappa = 1$, we substitute $\alpha = M/|R| - 1$ and reach

$$m_i(y) = \frac{1}{M-|R|}\sum_{j\notin R}\frac{\hat{p}(y)}{|R|\,\hat{p}(y)+(M-|R|)\hat{p}_{\leftarrow j}(y)} \quad \text{if } i \in R \tag{B.103}$$

$$m_i(y) = \frac{\hat{p}_{\leftarrow i}(y)}{|R|\,\hat{p}(y)+(M-|R|)\,\hat{p}_{\leftarrow i}(y)} \quad \text{if } i \notin R, \tag{B.104}$$

which we call *uniform* Pairwise MIS.

• **The defensive case.** If we instead treat the canonical samples as more reliable than the other samples, we may interpolate the previous solution towards always choosing one of the canonical samples by keeping $\alpha = M/|R| - 1$ and choosing $0 \leq \kappa < 1$. One such a heuristic could be to ensure that the MIS weights of the canonical samples are always at least as large as those of the other samples. With $\alpha = M/|R| - 1$, the canonical $m_i(y)$ cannot be less than $(1-\kappa)/|R|$ (set $\hat{p}_j(y) \to \infty$ in Equation B.96), and the non-canonical

m_i cannot exceed $\kappa/(M-|R|)$ (set $\hat{p}_{\leftarrow i}(y) \to \infty$ in Equation B.97). These bounds can be made equal by choosing $(1-\kappa)/|R| = \kappa/(M-|R|)$, i.e., $\kappa = (M-|R|)/M$, which gives us the *defensive* generalized Pairwise MIS weights,

$$m_i(y) = \frac{1}{M} + \frac{1}{M}\sum_{j\notin R} \frac{\hat{p}(y)}{|R|\,\hat{p}(y) + (M-|R|)\,\hat{p}_{\leftarrow j}(y)} \quad \text{if } i \in R \tag{B.105}$$

$$m_i(y) = \frac{M-|R|}{M} \frac{\hat{p}_{\leftarrow i}(y)}{|R|\,\hat{p}(y) + (M-|R|)\,\hat{p}_{\leftarrow i}(y)} \quad \text{if } i \notin R. \tag{B.106}$$

B.4.3 Resampling Weight Bounds

We next derive more accurate bounds for the resampling weights for our generalized Talbot and Pairwise MIS weights. In all cases we achieve the same bound, $C_i/|R|$, where $\hat{p}_i(X_i)W_i \le C_i$.

• **Generalized Talbot MIS.** A direct substitution of the Generalized Talbot MIS weights (Equation B.90) into the formula of w_i (Equation B.84) yields, assuming that Y_i exists (otherwise $w_i = 0$),

$$\begin{aligned}
w_i &= m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right| && \text{(B.107)}\\
&= \left(\frac{\hat{p}_{\leftarrow i}(Y_i)}{\sum_{j=1}^{M} \hat{p}_{\leftarrow j}(Y_i)}\right) \cdot \hat{p}(Y_i) W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right| && \text{(B.108)}\\
&= \frac{\hat{p}_i(X_i)\left|\frac{\partial T_i^{-1}}{\partial Y_i}\right|}{\sum_{j\in R} \hat{p}_{\leftarrow j}(Y_i) + \sum_{j\notin R} \hat{p}_{\leftarrow j}(Y_i)} \cdot \hat{p}(Y_i) W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right| && \text{(B.109)}\\
&= \frac{\hat{p}(Y_i)}{|R|\,\hat{p}(Y_i) + \sum_{j\notin R} \hat{p}_{\leftarrow j}(Y_i)} \cdot \hat{p}_i(X_i) W_i && \text{(B.110)}\\
&\le \frac{1}{|R|} \cdot C_i = \frac{C_i}{|R|}. && \text{(B.111)}
\end{aligned}$$

• **Generalized Pairwise MIS.** We first cover the feasible parameter combinations $\alpha > 0$ and $0 \le \kappa \le 1$ in one go, assuming that $\hat{p}_i(X_i)W_i \le C_i$. For canonical samples $i \in R$, we have $\hat{p}_i = \hat{p}$ and $Y_i = T_i(X_i) = X_i$, we substitute Equation B.96 and get the bound

$$w_i = m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial Y_i}{\partial X_i} \right| \tag{B.112}$$

$$= \frac{1}{|R|} \left(1 - \kappa + \frac{\kappa}{M - |R|} \sum_{j \notin R} \frac{\hat{p}(Y_i)}{\hat{p}(Y_i) + \alpha \hat{p}_{\leftarrow j}(Y_i)} \right) \cdot \hat{p}(Y_i) W_i \cdot 1 \tag{B.113}$$

$$\leq \frac{1}{|R|} \left(1 - \kappa + \frac{\kappa}{M - |R|} \sum_{j \notin R} \frac{\hat{p}(Y_i)}{\hat{p}(Y_i)} \right) \cdot \hat{p}(Y_i) W_i \tag{B.114}$$

$$\leq \frac{C_i}{|R|}. \tag{B.115}$$

For non-canonical samples $i \notin R$, assuming again $\hat{p}_i(X_i) W_i \leq C_i$, substituting Equation B.97, we get the bound

$$w_i = m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial Y_i}{\partial X_i} \right| \tag{B.116}$$

$$= \left(\alpha \frac{\kappa}{M - |R|} \frac{\hat{p}_{\leftarrow i}(Y_i)}{\hat{p}(Y_i) + \alpha \hat{p}_{\leftarrow i}(Y_i)} \right) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial Y_i}{\partial X_i} \right| \tag{B.117}$$

$$= \alpha \frac{\kappa}{M - |R|} \frac{\hat{p}_i(X_i)}{\hat{p}(Y_i) + \alpha \hat{p}_{\leftarrow i}(Y_i)} \cdot \hat{p}(Y_i) W_i \tag{B.118}$$

$$\leq \alpha \frac{\kappa}{M - |R|} C_i. \tag{B.119}$$

$$\tag{B.120}$$

In the case that $\alpha \leq M / |R| - 1$, we simplify

$$w_i \leq \kappa \frac{C_i}{|R|} \leq \frac{C_i}{|R|}. \tag{B.121}$$

B.4.4 Tractable Marginal PDFs

Sometimes the PDFs of the input samples X_i are tractable functions p_i. In that case, the PDFs p_i may be used in place of the $\hat{p}_i$ in the MIS weight formulas, effectively replacing $\hat{p}_{\leftarrow i}$ with the following "p from i":

$$p_{\leftarrow i}(y) = \begin{cases} p_i \left(T_i^{-1}(y) \right) \left| T_i^{-1\prime} \right| (y), & \text{if } y \in \mathcal{D}(T_i^{-1}) \\ 0 & \text{otherwise} \end{cases}, \tag{B.122}$$

resulting in the following expression for the generalized Talbot MIS:

$$m_i(y) = \frac{p_{\leftarrow i}(Y_i)}{\sum_{j=1}^{M} p_{\leftarrow j}(Y_i)}. \tag{B.123}$$

The Pairwise MIS expression additionally contains terms $\hat{p}(y)$ whose normalization may differ significantly from that of the PDFs p_i. As such, we suggest replacing the terms

$\hat{p}(y)$ in the MIS with a fixed canonical importance sampler $c \in C$ that is reasonable for integrating $\hat{p}$ $(\hat{p}(y) \leq C_c p_c(y))$. We show the uniform case as an example of this translation to known PDFs:

$$m_i(y) = \frac{1}{M - |R|} \sum_{j \notin R} \frac{p_c(y)}{|R|\, p_c(y) + (M - |R|) p_{\leftarrow j}(y)} \quad \text{if } i \in R \tag{B.124}$$

$$m_i(y) = \frac{\hat{p}_{\leftarrow i}(y)}{|R|\, p_c(y) + (M - |R|)\, p_{\leftarrow i}(y)} \quad \text{if } i \notin R. \tag{B.125}$$

We then derive the resampling weight bounds for these updated formulas. Since the p_i are tractable, we assume unbiased contribution weights $W_i = 1/p_i(X_i)$. We also assume that the canonical samples are reasonably importance sampled for $\hat{p}$, i.e., $\hat{p}(x) \leq C_i p_i(x)$ for all $i \in R$.

• **Talbot MIS.** Substituting Equation B.123 into Equation B.84, yields, remembering that $p_{\leftarrow j}(y) = p_j(y)$ for canonical j,

$$w_i = m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial T_i}{\partial X_i} \right| \tag{B.126}$$

$$= \left(\frac{p_{\leftarrow i}(Y_i)}{\sum_{j=1}^{M} p_{\leftarrow j}(Y_i)} \right) \cdot \frac{\hat{p}(Y_i)}{p_i(X_i)} \cdot \left| \frac{\partial T_i}{\partial X_i} \right| \tag{B.127}$$

$$= \frac{p_i(X_i)}{\sum_{j \in R} p_{\leftarrow j}(Y_i) + \sum_{j \notin R} p_{\leftarrow j}(Y_i)} \cdot \frac{\hat{p}(Y_i)}{p_i(X_i)} \tag{B.128}$$

$$= \frac{\hat{p}(Y_i)}{\sum_{j \in R} p_j(Y_i) + \sum_{j \notin R} p_{\leftarrow j}(Y_i)} \tag{B.129}$$

$$\leq \frac{\hat{p}(Y_i)}{\sum_{j \in R} p_j(Y_i)} \leq \frac{\hat{p}(Y_i)}{|R| \min_{j \in R} p_j(Y_i)} \tag{B.130}$$

$$= \frac{1}{|R|} \max_{j \in R} \frac{\hat{p}(Y_i)}{p_j(Y_i)} \leq \frac{1}{|R|} \max_{j \in R} C_j. \tag{B.131}$$

• **Pairwise MIS.** We now derive bounds for the resampling weights in the case of Generalized Pairwise MIS weights with $0 \leq \alpha \leq M/|R| - 1$ and $0 \leq \kappa \leq 1$, using p_i instead of $\hat{p}_i$.

If i is a canonical index, we use Equation B.96 with $\hat{p}_{\leftarrow j}$ replaced with $p_{\leftarrow j}$ and $\hat{p}$ replaced with p_c. Noting that for canonical indices $Y_i = T_i(X_i) = X_i$, we reach

$$
\begin{aligned}
w_i &= m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial T_i}{\partial X_i} \right| && \text{(B.132)} \\
&= \frac{1}{|R|} \left(1 - \kappa + \frac{\kappa}{M - |R|} \sum_{j \notin R} \frac{p_c(Y_i)}{p_c(Y_i) + \alpha p_{\leftarrow j}(Y_i)} \right) \cdot \frac{\hat{p}(Y_i)}{p_i(X_i)} \cdot 1 && \text{(B.133)} \\
&\le \frac{1}{|R|} \left(1 - \kappa + \frac{\kappa}{M - |R|} \sum_{j \notin R} 1 \right) \cdot \frac{\hat{p}(X_i)}{p_i(X_i)} && \text{(B.134)} \\
&= \frac{1}{|R|} \cdot \frac{\hat{p}(X_i)}{p_i(X_i)} \le \frac{C_i}{|R|}. && \text{(B.135)}
\end{aligned}
$$

Similarly, for non-canonical i we use Equation B.97 with the same substitutions and reach

$$
\begin{aligned}
w_i &= m_i(Y_i) \cdot \hat{p}(Y_i) W_i \cdot \left| \frac{\partial T_i}{\partial X_i} \right| && \text{(B.136)} \\
&= \left(\alpha \frac{\kappa}{M - |R|} \frac{p_{\leftarrow i}(Y_i)}{p_c(Y_i) + \alpha p_{\leftarrow i}(Y_i)} \right) \cdot \frac{\hat{p}(Y_i)}{p_i(X_i)} \cdot \left| \frac{\partial T_i}{\partial X_i} \right| && \text{(B.137)} \\
&= \left(\alpha \frac{\kappa}{M - |R|} \frac{p_i(X_i) \left| \frac{\partial T_i^{-1}}{\partial Y_i} \right|}{p_c(Y_i) + \alpha p_{\leftarrow i}(Y_i)} \right) \cdot \frac{\hat{p}(Y_i)}{p_i(X_i)} \cdot \left| \frac{\partial T_i}{\partial X_i} \right| && \text{(B.138)} \\
&= \alpha \frac{\kappa}{M - |R|} \frac{\hat{p}(Y_i)}{p_c(Y_i) + \alpha p_{\leftarrow i}(Y_i)} \le \alpha \frac{\kappa}{M - |R|} \frac{\hat{p}(Y_i)}{p_c(Y_i)} && \text{(B.139)} \\
&\le \alpha \frac{\kappa}{M - |R|} C_c \le \left(\frac{M}{|R|} - 1 \right) \frac{1}{M - |R|} C_c = \frac{C_c}{|R|}. && \text{(B.140)}
\end{aligned}
$$

We can combine the above results into a single, slightly looser bound that works for any i and both MIS weight families (Talbot and Pairwise) when used with tractable PDFs:

$$
w_i \le \frac{1}{|R|} \max_{j \in R} C_j. \tag{B.141}
$$

B.5 Correctness Notes

This section discusses some aspects of ReSTIR that may affect performance and correctness.

• **On Visibility.** Relating to the discussion in Section 6.5.1, ReSTIR DI [7] used a target $\hat{p}_i$ without the visibility term $V(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$. Our ReSTIR PT always considers visibility between vertices. In fact, neglecting visibility makes maintaining convergence guarantees tricky, as it creates paths with positive $\hat{p}_i$ that never get sampled as X_i due to occlusion. This implies $\operatorname{supp} \hat{p}_i \not\subset \operatorname{supp} X_i$, making X_i non-canonical.

Without extra guarantees, Y resampled from X_i no longer covers $\operatorname{supp} \hat{p}_i$, breaking the $\operatorname{supp} \hat{p} \subset \operatorname{supp} Y_M$ assumption of Equation 6.23 and preventing convergence of p_Y to $\bar{p}$.

ReSTIR DI circumvents this by closely tracking reuse throughout each algorithmic phase. While its $\hat{p}_i$ for temporal reuse checks for visibility, its spatial reuse uses an unoccluded target $\hat{p}_i^{-V}$ to reduce cost. Without full coverage of $\hat{p}_i^{-V}$'s domain, intermediate distributions never converge. The design still ensures coverage of f_i's domain, allowing *final* estimators to remain unbiased. For direct lighting, earlier convergence to $\hat{p}_i$ may only be of theoretic interest, with the choices by Bitterli et al. [7] working around a bottleneck from visibility costs. In path tracing, ignoring visibility requires much more engineering, as typical path samplers never select occluded paths with $V(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) = 0$.

• **Temporal Reuse.** Temporal sample reuse is unbiased with proper MIS weights, e.g., our generalized Talbot or pairwise MIS, but a temporal shift mapping is needed, as evaluating MIS weights for GRIS requires bijectively shifting paths between the prior and current frames. In some cases, e.g., conflicting motion vectors, careful map definitions may be needed to retain bijectivity.

This constraint means fully unbiased temporal reuse must evaluate paths in both the current and prior frames, which is tricky in dynamic environments. Biased approximations to temporal MIS can be used, e.g., neglecting visibility [7, 10], which gives desirable performance improvements for often imperceptible bias. Lin et al. [157] explicitly account for temporal changes, reporting it reduces response time to dynamic lighting.

B.6 Primary Sample Space

Performing integration in a Monte Carlo setting typically starts from primary sample sequences, i.e., streams of random numbers $\bar{\mathbf{U}} = (U_1, U_2, \ldots)$, where $U_i \in [0, 1)$ are uniformly distributed. Each $\bar{\mathbf{U}}$ is used to estimate a contribution $F(\bar{\mathbf{U}})$, and the Monte Carlo integration result is

$$I = \mathbb{E}\left[F(\bar{\mathbf{U}})\right] = \int_{\mathscr{U}} F(\bar{\mathbf{u}})\, \mathrm{d}\bar{\mathbf{u}}. \tag{B.142}$$

A unidirectional path tracer builds a sequence of paths of different lengths from the random sequences $\bar{\mathbf{U}}$. Often, the path tracer produces, for each length d, N paths $X_{d,n} \in \Omega_d$ by different strategies. Here, Ω_d is the space of all paths of length d. N is often 2, and the paths $X_{d,n}$ with different n correspond to a next-event-estimation path connected to a random light, and path continued to a direction importance sampled according to the BSDF. The paths $X_{d,n}$ are functions of $\bar{\mathbf{U}}$, i.e., $X_{d,n} = x_{d,n}(\bar{\mathbf{U}})$.

The total path contribution is the sum of the integrals of the fixed-length path contributions,

$$I = \sum_{d=1}^{\infty} \int_{\Omega_d} f(x)\,\mathrm{d}x. \tag{B.143}$$

We factor in the MIS weights $\omega_{d,n}$ of the n sampling strategies and reach

$$I = \sum_{d=1}^{\infty} \int_{\Omega_d} \left[\sum_{n=1}^{N} \omega_{d,n}(x)\right] f(x)\,\mathrm{d}x \tag{B.144}$$

$$= \sum_{d=1}^{\infty} \sum_{n=1}^{N} \int_{\Omega_d} \omega_{d,n}(x) f(x)\,\mathrm{d}x. \tag{B.145}$$

We assume for each term a proper importance sampler that produces paths $X_{d,n} \in \Omega_d$ with density $p_{d,n}$. We assume that $\omega_{d,n}(x) = 0$ whenever $p_{d,n}(x) = 0$ to retain the partition of unity. This yields us

$$I = \sum_{d=1}^{\infty} \sum_{n=1}^{N} \mathbb{E}\left[\omega_{d,n}(X_{d,n}) \frac{f(X_{d,n})}{p_{d,n}(X_{d,n})}\right]. \tag{B.146}$$

Since the $X_{d,n}$ are generated from the random variables $\bar{\mathbf{U}}$ by $X_{d,n} = x_{d,n}(\bar{\mathbf{U}})$, we may write

$$I = \sum_{d=1}^{\infty} \sum_{n=1}^{N} \mathbb{E}\left[\omega_{d,n}\left(x_{d,n}(\bar{\mathbf{U}})\right) \frac{f\left(x_{d,n}(\bar{\mathbf{U}})\right)}{p_{d,n}\left(x_{d,n}(\bar{\mathbf{U}})\right)}\right]. \tag{B.147}$$

The fact that many $\bar{\mathbf{U}}$ may lead to the same path $X_{d,n}$ does not complicate this fact. We then write the expectations as integrals and reach

$$I = \sum_{d=1}^{\infty} \sum_{n=1}^{N} \int_{\mathscr{U}} \omega_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right) \frac{f\left(x_{d,n}(\bar{\mathbf{u}})\right)}{p_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right)}\,\mathrm{d}\bar{\mathbf{u}} \tag{B.148}$$

$$= \int_{\mathscr{U}} \sum_{d=1}^{\infty} \sum_{n=1}^{N} \omega_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right) \frac{f\left(x_{d,n}(\bar{\mathbf{u}})\right)}{p_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right)}\,\mathrm{d}\bar{\mathbf{u}}. \tag{B.149}$$

This yields us the F for Equation B.142,

$$F(\bar{\mathbf{u}}) = \sum_{d=1}^{\infty} \sum_{n=1}^{N} \omega_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right) \frac{f\left(x_{d,n}(\bar{\mathbf{u}})\right)}{p_{d,n}\left(x_{d,n}(\bar{\mathbf{u}})\right)}. \tag{B.150}$$

B.7 Parameter Exploration

In this section, we analyze the effect of parameters such as neighbor count, reuse window size and the number of spatial reuse passes for ReSTIR PT for offline rendering, justifying our choice of default parameters. We perform the experiments leading to our conclusions in two scenes, the simple Cornell Box and the more complex Kitchen scene.

We first demonstrate how sampling cost can be amortized by increasing the number of pixels we reuse from and the number of candidate path samples per pixel into optimal numbers. For the first experiment, we densely reuse paths from each pixel in a square around the current pixel, and after analyzing this case, we generalize the results to randomly sampled sparse neighbors from a larger neighborhood.

B.7.1 Parameters for Dense Block of Pixels

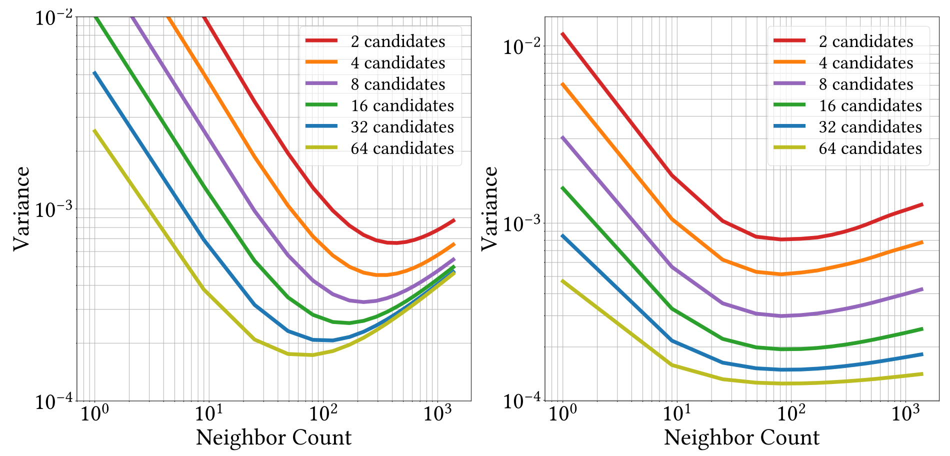
Path reuse without GRIS (e.g., the path reuse algorithm by Bekaert et al. [11]) can already achieve higher sampling efficiency than pure path tracing because resampling is cheaper than generating samples from scratch. By defining unbiased contribution weights, our GRIS supports more aggressive amortization of the sampling cost–with a reservoir, we can increase the number of input samples for the initial RIS to further amortize the sample generation cost. For S candidate samples, we approximately "gain" S samples when reusing one sample. To analyze the sampling efficiency, we form a simplified model that measures the ratio between the number of rays gained and the number of rays computed. Assuming a simplified ideal case where all pixels generate paths with a fixed length L and reusing a neighbor effectively gains all samples it generates, we write the following equation for reusing K pixels (including self),

$$\frac{\#\text{ rays gained}}{\#\text{ rays computed}} = \frac{KSL}{SL + \eta(K-1)}\,, \tag{B.151}$$

where η is the ray cost we pay for resampling (usually $\eta < L$).

If $S = 1$, the equation evaluates to $KL/(L + \eta(K-1))$. Even when $K \to \infty$, the efficiency is still bounded by L/η. If we have $S \to \infty$, the efficiency is bounded by K, which means that the efficiency improvement becomes theoretically unlimited in this simplified ideal case. In practice, we can measure sampling efficiency by comparing the variance at equal render time.

This suggests that we should use a relatively large S and a relatively large K to achieve higher sampling efficiency. In practice, we observe in Figure B.1 that increasing S and K eventually becomes harmful: increasing the neighbor count K increases MSE instead of reducing it when the neighborhood becomes large enough, and a larger S leads to problems even sooner. This is because enlarging the neighborhood size adds samples that are farther away, and path space similarity generally decreases by distance. This eventually offsets



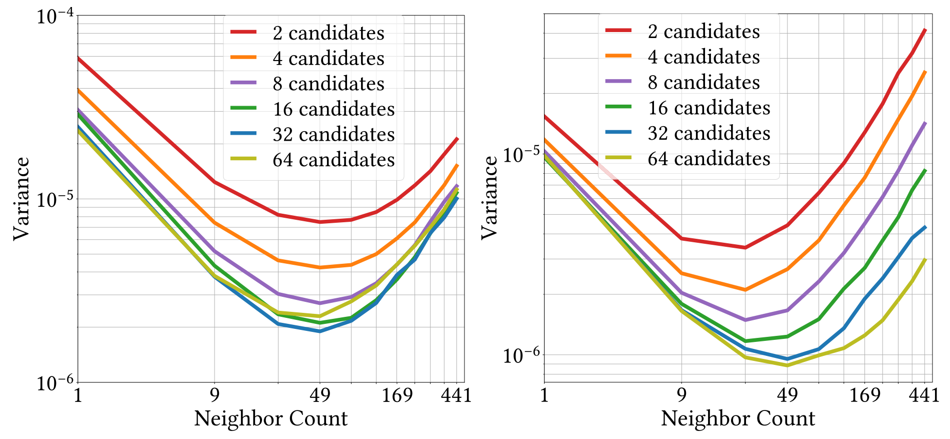
(a) Cornell Box (b) Kitchen

Figure B.1: A parameter exploration of the number of initial samples per pixel (S, colored lines) and the number of neighbors used for resampling (X-axis). Each pixel generates S candidate samples and uses RIS to select one. Then GRIS resamples for each pixel a path from one from the neighboring 3x3, 5x5, 7x7, etc. pixels. Pairwise MIS is used.

the benefit from more samples. Enlarging S (the candidate count for initial resampling) can also eventually lead to diminishing returns, as we show in Figure B.2. In both scenes, we see that using a combination of $S = 32$ and $K = 49$ (i.e., a 7×7 neighborhood) results in near-optimal sampling efficiency.

B.7.2 Parameters for Sparse Neighbors

The cost of using a large number of input samples for GRIS can be amortized also by chaining multiple spatial reuse passes. A caveat is that chaining spatial reuse passes

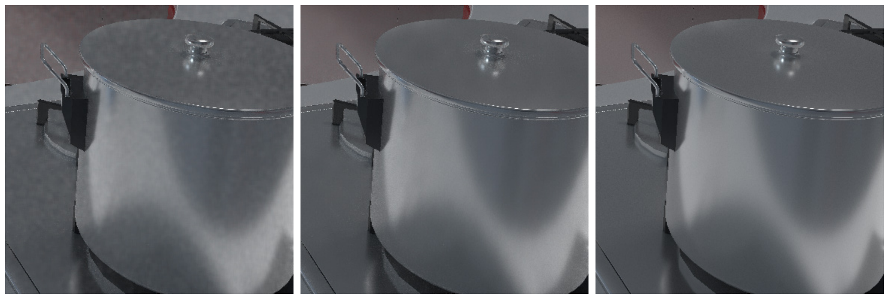


(a) Cornell Box (b) Kitchen

Figure B.2: Sampling efficiency. This experiment repeats the setting of Figure B.1, but outputs the MSE after ten seconds of rendering, revealing the most efficient parameters for multisample rendering.

with a small, fixed neighborhood can lead to excessive correlation between the samples, which can lead to reduced sampling efficiency. This motivates using sparse neighbors randomized from a larger neighborhood to minimize correlation.

From our parameter exploration, we found that near-optimal sampling efficiency for random reuse can be achieved with $S = 32$ (similar to the dense reuse case), 2-3 rounds of spatial reuse with 6-10 random neighbors in 5-10 pixel radius (a diameter of 10-20 pixels). After comparing visual quality in this parameter range for both scenes, we select a parameter set of 10 pixel radius, 3 rounds of spatial reuse, and 6 random neighbors. We see a slight variance reduction in equal render time compared to dense reuse with its near-optimal parameters. The improvement of sampling efficiency is small, because sparse reuse requires a larger neighborhood to reduce correlation, which also lowers the similarity between the pixels. This partially cancels the benefits from amortizing the sampling cost. The visual improvement is, however, much larger (Figure B.3), since random reuse reduces visual correlation between nearby pixels. We find the 10-pixel radius can still be enlarged for real-time rendering, as chaining many reuse passes over multiple frames builds up more correlation. For real-time rendering, we use a radius of 20 pixels, only one initial candidate sample per pixel, and one spatial reuse pass between the current pixel and three random others, to keep the rendering time low.



(a) Dense Reuse
MSE: 2.45e-7
MAPE: 0.0571

(b) Random Reuse
MSE: 2.37e-7
MAPE: 0.0502

(c) Reference

Figure B.3: A comparison of 60-second equal-time rendering of the Kitchen scene. Dense reuse and random reuse have similar MSE/MAPE with their respective optimal parameters, but random reuse produces results visually closer to the reference.

REFERENCES

[1] M. McGuire, “Computer graphics archive,” July 2017. [Online]. Available: https://casual-effects.com/data

[2] C. Schied, A. Kaplanyan, C. Wyman, A. Patney, C. R. A. Chaitanya, J. Burgess, S. Liu, C. Dachsbacher, A. Lefohn, and M. Salvi, “Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination,” in *Proceedings of High Performance Graphics*. ACM, 2017, p. 2.

[3] P. Moreau, M. Pharr, and P. Clarberg, “Dynamic Many-Light Sampling for Real-Time Ray Tracing,” in *High-Performance Graphics - Short Papers*, M. Steinberger and T. Foley, Eds. The Eurographics Association, 2019.

[4] C. Yuksel, “Stochastic lightcuts,” in *High-Performance Graphics (HPG 2019)*. The Eurographics Association, 2019.

[5] P. Kutz, R. Habel, Y. K. Li, and J. Novák, “Spectral and decomposition tracking for rendering heterogeneous volumes,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–16, 2017.

[6] J. Novák, A. Selle, and W. Jarosz, “Residual ratio tracking for estimating attenuation in participating media.” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, pp. 179–1, 2014.

[7] B. Bitterli, C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz, “Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 148:1–148:17, 2020.

[8] J. F. Talbot, “Importance resampling for global illumination,” Master’s thesis, Brigham Young University, 2005.

[9] B. Miller, I. Georgiev, and W. Jarosz, “A null-scattering path integral formulation of light transport,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–13, 2019.

[10] Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni, “ReSTIR GI: Path resampling for real-time path tracing,” *Computer Graphics Forum*, vol. 40, no. 8, pp. 17–29, 2021.

[11] P. Bekaert, M. Sbert, and J. H. Halton, “Accelerating path tracing by re-using paths,” in *Rendering Techniques*, 2002, pp. 125–134.

[12] NVIDIA, “Nvidia real-time denoisers,” https://developer.nvidia.com/nvidia-rt-denoiser, 2021, [Online; accessed 9-March-2022].

[13] E. Kilgariff, H. Moreton, N. Stam, and B. Bell, “NVIDIA Turing architecture in-

depth," https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/, 2018, [Online; accessed 9-December-2021].

[14] NVIDIA, "NVIDIA® OptiX™ AI-Accelerated Denoiser," Nov. 2017. [Online]. Available: https://developer.nvidia.com/optix-denoiser

[15] R. Cowgill, "Introducing ray tracing in unreal engine 4," https://developer.nvidia.com/blog/introducing-ray-tracing-in-unreal-engine-4/, 2020, [Online; accessed 9-March-2022].

[16] C. Yuksel and C. Yuksel, "Lighting grid hierarchy for self-illuminating explosions," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 110, 2017.

[17] J. Talbot, D. Cline, and P. Egbert, "Importance Resampling for Global Illumination," in *Eurographics Symposium on Rendering (2005)*, K. Bala and P. Dutre, Eds. The Eurographics Association, 2005.

[18] J. Hasselgren, J. Munkberg, M. Salvi, A. Patney, and A. Lefohn, "Neural Temporal Adaptive Sampling and Denoising," *Computer Graphics Forum*, 2020.

[19] C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila, "Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, jul 2017.

[20] C. Schied, C. Peters, and C. Dachsbacher, "Gradient estimation for real-time adaptive temporal filtering," *Proceedings of the ACM in Computer Graphics and Interactive Techniques*, vol. 1, no. 2, 2018.

[21] P. Bauszat, V. Petitjean, and E. Eisemann, "Gradient-domain path reusing," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, nov 2017.

[22] E. Heitz, S. Hill, and M. McGuire, "Combining analytic direct illumination and stochastic shadows," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2018, p. 2.

[23] T. Stachowiak, "Stochastic screen-space reflections," in *Advances in Real Time Rendering, (ACM SIGGRAPH Courses)*, 2015.

[24] P. Shirley, C. Wang, and K. Zimmerman, "Monte carlo techniques for direct lighting calculations," *ACM Transactions on Graphics (TOG)*, vol. 15, no. 1, pp. 1–36, 1996.

[25] C. Peters, "Brdf importance sampling for polygonal lights," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, jul 2021.

[26] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg, "Lightcuts: a scalable approach to illumination," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1098–1107, 2005.

[27] H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*. USA: A. K. Peters, Ltd., 2001.

[28] C. Dachsbacher, J. Křivánek, M. Hašan, A. Arbree, B. Walter, and J. Novák, "Scalable realistic rendering with many-light methods," *Computer Graphics Forum*, vol. 33, no. 1, pp. 88–104, 2014.

[29] G. J. Ward, "Adaptive shadow testing for ray tracing," in *Photorealistic Rendering in Computer Graphics*. Springer, 1994, pp. 11–20.

[30] E. Paquette, P. Poulin, and G. Drettakis, "A light hierarchy for fast rendering of scenes with many lights," *Computer Graphics Forum*, vol. 17, no. 3, pp. 63–74, 1998.

[31] S. Fernandez, K. Bala, and D. P. Greenberg, "Local illumination environments for direct lighting acceleration." *Rendering Techniques*, vol. 2002, p. 13th, 2002.

[32] A. Keller, "Instant radiosity," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 49–56.

[33] H. Rehfeld and C. Dachsbacher, "Lightcut interpolation," in *Proceedings of High Performance Graphics*, 2016, pp. 99–108.

[34] B. Walter, A. Arbree, K. Bala, and D. P. Greenberg, "Multidimensional lightcuts," *ACM Transactions on Graphics (Proceedings of SIGGRAPH '06)*, vol. 25, no. 3, pp. 1081–1088, 2006.

[35] T. Davidovič, I. Georgiev, and P. Slusallek, "Progressive lightcuts for gpu," in *ACM SIGGRAPH 2012 Talks*. ACM, 2012, p. 1.

[36] B. Walter, P. Khungurn, and K. Bala, "Bidirectional lightcuts," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 59:1–59:11, 2012.

[37] R. Wang, Y. Huo, Y. Yuan, K. Zhou, W. Hua, and H. Bao, "Gpu-based out-of-core many-lights rendering," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 210:1–210:10, 2013.

[38] P. Vévoda and J. Křivánek, "Adaptive direct illumination sampling," in *SIGGRAPH ASIA 2016 Posters*. ACM, 2016, p. 43.

[39] P. Vévoda, I. Kondapaneni, and J. Křivánek, "Bayesian online regression for adaptive direct illumination sampling," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 125, 2018.

[40] A. Keller, C. Wächter, M. Raab, D. Seibert, D. van Antwerpen, J. Korndörfer, and L. Kettner, "The iray light transport simulation and rendering system," in *ACM SIGGRAPH 2017 Talks*. ACM, 2017, p. 34.

[41] A. C. Estevez and C. Kulla, "Importance sampling of many lights with adaptive tree splitting," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 2, p. 25, 2018.

[42] P. Moreau and P. Clarberg, "Importance sampling of many lights on the gpu," in *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*,

E. Haines and T. Akenine-Möller, Eds. Berkeley, CA: Apress, 2019, pp. 255–283.

[43] M. Hašan, F. Pellacini, and K. Bala, "Matrix row-column sampling for the many-light problem," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 26, 2007.

[44] T. Davidovic, J. Krivnek, M. Hasan, P. Slusallek, and K. Bala, "Combining global and local lights for high-rank illumination effects," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 5, 2010.

[45] M. Hašan, E. Velázquez-Armendariz, F. Pellacini, and K. Bala, "Tensor clustering for rendering many-light animations," in *Proceedings of Eurographics Workshop on Rendering*, 2008, pp. 1105–1114.

[46] J. Ou and F. Pellacini, "Lightslice: matrix slice sampling for the many-lights problem." *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, pp. 179–1, 2011.

[47] Y. Huo, R. Wang, S. Jin, X. Liu, and H. Bao, "A matrix sampling-and-recovery approach for many-lights rendering," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 210, 2015.

[48] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[49] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg, "The irradiance volume," *IEEE Computer Graphics and Applications*, vol. 18, no. 2, pp. 32–43, 1998.

[50] A. Kaplanyan and C. Dachsbacher, "Cascaded light propagation volumes for real-time indirect illumination," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, 2010, pp. 99–107.

[51] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive indirect illumination using voxel cone tracing," in *Computer Graphics Forum*, vol. 30, no. 7. Wiley Online Library, 2011, pp. 1921–1930.

[52] P. Christensen, "Point-based approximate color bleeding," *Pixar Technical Notes*, vol. 2, no. 5, p. 6, 2008.

[53] T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher, "Micro-rendering for scalable, parallel final gathering," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 132.

[54] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo, "Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 977–986, 2006.

[55] P.-P. Sloan, N. K. Govindaraju, D. Nowrouzezahrai, and J. Snyder, "Image-based proxy accumulation for real-time soft global illumination," in *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*. IEEE, 2007, pp. 97–105.

[56] H. W. Jensen, "Global illumination using photon maps," in *Rendering Techniques*.

Springer, 1996, pp. 21–30.

[57] T. Hachisuka, S. Ogaki, and H. W. Jensen, "Progressive photon mapping," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 130.

[58] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche, "Bidirectional instant radiosity." in *Rendering Techniques*, 2006, pp. 389–397.

[59] R. Ramamoorthi and P. Hanrahan, "An efficient representation for irradiance environment maps," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 2001, pp. 497–500.

[60] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 527–536.

[61] P. Green, J. Kautz, and F. Durand, "Efficient reflectance and visibility approximations for environment map rendering," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 495–502.

[62] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating dynamic global illumination in image space," in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 2009, pp. 75–82.

[63] G. Nichols, J. Shopf, and C. Wyman, "Hierarchical image-space radiosity for interactive global illumination," in *Computer Graphics Forum*, vol. 28, no. 4. Wiley Online Library, 2009, pp. 1141–1149.

[64] M. McGuire and M. Mara, "Efficient gpu screen-space ray tracing," *Journal of Computer Graphics Techniques (JCGT)*, vol. 3, no. 4, pp. 73–85, 2014.

[65] M. Mara, M. McGuire, D. Nowrouzezahrai, and D. P. Luebke, "Deep g-buffers for stable global illumination approximation." in *High Performance Graphics*, 2016, pp. 87–98.

[66] P. Moreau, E. Sintorn, V. Kämpe, U. Assarsson, and M. C. Doggett, "Photon splatting using a view-sample cluster hierarchy," in *High Performance Graphics*, 2016, pp. 75–85.

[67] K. Vardis, G. Papaioannou, and A. Gkaravelis, "Real-time radiance caching using chrominance compression," *Journal of Computer Graphics Techniques*, vol. 3, no. 4, 2014.

[68] J. Jendersie, D. Kuri, and T. Grosch, "Precomputed illuminance composition for real-time global illumination," in *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2016, pp. 129–137.

[69] J. Krivanek, P. Gautron, S. Pattanaik, and K. Bouatouch, "Radiance caching for efficient global illumination computation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 5, pp. 550–561, 2005.

[70] A. Silvennoinen and J. Lehtinen, "Real-time global illumination by precomputed

local reconstruction from sparse radiance probes," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 230, 2017.

[71] M. McGuire, M. Mara, D. Nowrouzezahrai, and D. Luebke, "Real-time global illumination using precomputed light field probes," in *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2017, p. 2.

[72] Z. Majercik, J.-P. Guertin, D. Nowrouzezahrai, and M. McGuire, "Dynamic diffuse global illumination with ray-traced irradiance fields," *Journal of Computer Graphics Techniques (JCGT)*, vol. 8, no. 2, pp. 1–30, June 2019.

[73] M. Wrenninge, C. D. Kulla, and V. Lundqvist, "Oz: the great and volumetric." in *SIGGRAPH Talks*, 2013, pp. 46–1.

[74] T.-Y. Kim and U. Neumann, "Opacity shadow maps," in *Rendering Techniques*. Springer, 2001, pp. 177–182.

[75] C. Yuksel and J. Keyser, "Deep opacity maps," *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008)*, vol. 27, no. 2, pp. 675–680, 2008.

[76] M. Salvi, K. Vidimče, A. Lauritzen, and A. Lefohn, "Adaptive volumetric shadow maps," *Computer Graphics Forum*, vol. 29, no. 4, pp. 1289–1296, 2010.

[77] J. Jansen and L. Bavoil, "Fourier opacity mapping," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010, pp. 165–172.

[78] C. Delalandre, P. Gautron, J.-E. Marvie, and G. François, "Transmittance function mapping," in *Symposium on Interactive 3D Graphics and Games*, 2011, pp. 31–38.

[79] P. Gautron, C. Delalandre, J.-E. Marvie, and P. Lecocq, "Boundary-aware extinction mapping," *Computer Graphics Forum*, vol. 32, no. 7, pp. 305–314, 2013.

[80] S. Hillaire, "Physically-based and unified volumetric rendering in frostbite," in *SIGGRAPH Courses: Advances in Real-Time Rendering in Games*, 2015.

[81] T. Sutton, F. Brown, F. Bischoff, D. MacMillan, C. Ellis, J. Ward, C. Ballinger, D. Kelly, and L. Schindler, "The physical models and statistical procedures used in the racer monte carlo code," Knolls Atomic Power Lab., Tech. Rep., 1999.

[82] J. Novák, I. Georgiev, J. Hanika, and W. Jarosz, "Monte Carlo methods for volumetric light transport simulation," *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)*, vol. 37, no. 2, May 2018.

[83] E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth, "Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry," in *Proceedings of the Conference on the Application of Computing Methods to Reactor Problems*, vol. 557, no. 2, 1965.

[84] M. Raab, D. Seibert, and A. Keller, "Unbiased global illumination with participating media," in *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 2008, pp. 591–605.

[85] L. Szirmay-Kalos, B. Tóth, and M. Magdics, "Free path sampling in high resolution inhomogeneous participating media," *Computer Graphics Forum*, vol. 30, pp. 85–97, March 2011.

[86] Y. Yue, K. Iwasaki, B. Chen, Y. Dobashi, and T. Nishita, "Toward optimal space partitioning for unbiased, adaptive free path sampling of inhomogeneous participating media," *Computer Graphics Forum*, vol. 30, 2011.

[87] M. Galtier, S. Blanco, C. Caliot, C. Coustet, J. Dauchet, M. El Hafi, V. Eymet, R. Fournier, J. Gautrais, A. Khuong *et al.*, "Integral formulation of null-collision monte carlo algorithms," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 125, pp. 57–68, 2013.

[88] L. Szirmay-Kalos, I. Georgiev, M. Magdics, B. Molnar, and D. Legrady, "Unbiased light transport estimators for inhomogeneous participating media," *Computer Graphics Forum*, vol. 36, pp. 9–19, May 2017.

[89] E. d'Eon and J. Novák, "Zero-variance Transmittance Estimation," in *Eurographics Symposium on Rendering - DL-only Track*, 2021.

[90] I. Georgiev, Z. Misso, T. Hachisuka, D. Nowrouzezahrai, J. Křivánek, and W. Jarosz, "Integral formulations of volumetric transmittance," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–17, 2019.

[91] M. Kettunen, E. D'Eon, J. Pantaleoni, and J. Novák, "An unbiased ray-marching transmittance estimator," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, Jul. 2021.

[92] L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the galaxy," *Astrophysical Journal*, vol. 93, pp. 70–83, 1941.

[93] F. Simon, J. Hanika, T. Zirr, and C. Dachsbacher, "Line integration for rendering heterogeneous emissive volumes," *Computer Graphics Forum*, vol. 36, pp. 101–110, July 2017.

[94] C. Kulla and M. Fajardo, "Importance sampling techniques for path tracing in participating media," *Computer graphics forum*, vol. 31, no. 4, pp. 1519–1528, 2012.

[95] I. Georgiev, J. Krivanek, T. Hachisuka, D. Nowrouzezahrai, and W. Jarosz, "Joint importance sampling of low-order volumetric scattering." *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 164–1, 2013.

[96] J. Křivánek and E. d'Eon, "A zero-variance-based sampling scheme for monte carlo subsurface scattering," in *ACM SIGGRAPH 2014 Talks*, 2014, pp. 1–1.

[97] J. Meng, J. Hanika, and C. Dachsbacher, "Improving the dwivedi sampling scheme," *Computer Graphics Forum*, vol. 35, no. 4, pp. 37–44, 2016.

[98] S. Herholz, Y. Zhao, O. Elek, D. Nowrouzezahrai, H. P. Lensch, and J. Křivánek, "Volume path guiding based on zero-variance random walk theory," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 3, pp. 1–19, 2019.

[99] H. W. Jensen and P. H. Christensen, “Efficient simulation of light transport in scenes with participating media using photon maps,” in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, 1998, pp. 311–320.

[100] W. Jarosz, M. Zwicker, and H. W. Jensen, “The beam radiance estimate for volumetric photon mapping,” in *ACM SIGGRAPH 2008 classes*, 2008, pp. 1–112.

[101] W. Jarosz, D. Nowrouzezahrai, I. Sadeghi, and H. W. Jensen, “A comprehensive theory of volumetric radiance estimation using photon points and beams,” *ACM Transactions on Graphics (TOG)*, vol. 30, no. 1, pp. 1–19, 2011.

[102] B. Bitterli and W. Jarosz, “Beyond points and beams: Higher-dimensional photon samples for volumetric light transport,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[103] X. Deng, S. Jiao, B. Bitterli, and W. Jarosz, “Photon surfaces for robust, unbiased volumetric density estimation,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 46, 2019.

[104] J. Novák, D. Nowrouzezahrai, C. Dachsbacher, and W. Jarosz, “Virtual ray lights for rendering scenes with participating media,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–11, 2012.

[105] ——, “Progressive virtual beam lights,” *Computer Graphics Forum*, vol. 31, no. 4, pp. 1407–1413, 2012.

[106] J. Křivánek, I. Georgiev, T. Hachisuka, P. Vévoda, M. Šik, D. Nowrouzezahrai, and W. Jarosz, “Unifying points, beams, and paths in volumetric light transport simulation,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–13, 2014.

[107] J. Lehtinen, T. Karras, S. Laine, M. Aittala, F. Durand, and T. Aila, “Gradient-domain metropolis light transport,” vol. 32, no. 4, Jul. 2013.

[108] M. Kettunen, M. Manzi, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker, “Gradient-domain path tracing,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, Jul. 2015.

[109] A. Gruson, B.-S. Hua, N. Vibert, D. Nowrouzezahrai, and T. Hachisuka, “Gradient-domain volumetric photon density estimation,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[110] J. Vorba, J. Hanika, S. Herholz, T. Müller, J. Křivánek, and A. Keller, “Path guiding in production,” in *ACM SIGGRAPH 2019 Courses*. New York, NY, USA: ACM, 2019, pp. 18:1–18:77.

[111] J. Boksansky, P. Jukarainen, and C. Wyman, “Rendering many lights with grid-based reservoirs,” in *Ray Tracing Gems II*. Springer, 2021, pp. 351–365.

[112] G. Boissé, “World-space spatiotemporal reservoir reuse for ray-traced global illumination,” in *SIGGRAPH Asia 2021 Technical Communications*, ser. SA ’21 Technical Communications. New York, NY, USA: Association for Computing Machinery,

2021.

[113] M. Manzi, M. Kettunen, F. Durand, M. Zwicker, and J. Lehtinen, "Temporal gradient-domain path tracing," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–9, 2016.

[114] B.-S. Hua, A. Gruson, V. Petitjean, M. Zwicker, D. Nowrouzezahrai, E. Eisemann, and T. Hachisuka, "A survey on gradient-domain rendering," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 455–472.

[115] M. Manzi, M. Kettunen, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker, "Gradient-domain bidirectional path tracing," 2015.

[116] B.-S. Hua, A. Gruson, D. Nowrouzezahrai, and T. Hachisuka, "Gradient-domain photon density estimation," in *Computer Graphics Forum*, vol. 36, no. 2. Wiley Online Library, 2017, pp. 31–38.

[117] W. Sun, X. Sun, N. A. Carr, D. Nowrouzezahrai, and R. Ramamoorthi, "Gradient-domain vertex connection and merging." in *EGSR (EI&I)*, 2017, pp. 83–92.

[118] V. Petitjean, P. Bauszat, and E. Eisemann, "Spectral gradient sampling for path tracing," in *Computer Graphics Forum*, vol. 37, no. 4. Wiley Online Library, 2018, pp. 45–53.

[119] D. Rubin, "A noniterative sampling/importance resampling alternative to data augmentation for creating a few imputations when fractions of missing information are modest: The sir algorithm," *Journal of the American Statistical Association*, vol. 82, pp. 544–546, 1987.

[120] A. Guetz, *Monte Carlo Methods for Structured Data*. Stanford University, 2012.

[121] M.-T. Chao, "A general purpose unequal probability sampling plan," *Biometrika*, vol. 69, no. 3, pp. 653–656, 1982.

[122] J. T. Kajiya, "The rendering equation," in *ACM Siggraph Computer Graphics*, vol. 20, no. 4. ACM, 1986, pp. 143–150.

[123] T. Kollig and A. Keller, "Illumination in the presence of weak singularities," in *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer, 2006, pp. 245–257.

[124] J. Novák, T. Engelhardt, and C. Dachsbacher, "Screen-space bias compensation for interactive high-quality global illumination with virtual point lights," in *Symposium on Interactive 3D Graphics and Games*. ACM, 2011, pp. 119–124.

[125] J. Sriwasansak, A. Gruson, and T. Hachisuka, "Efficient energy-compensated vpls using photon splatting," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1, p. 16, 2018.

[126] O. Olsson, M. Billeter, and U. Assarsson, "Clustered deferred and forward shading," in *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*. Eurographics Association, 2012, pp. 87–96.

[127] Y. Tokuyoshi and T. Harada, “Stochastic light culling,” *Journal of Computer Graphics Techniques Vol*, vol. 5, no. 1, 2016.

[128] O. Olsson and U. Assarsson, “Tiled shading,” *Journal of Graphics, GPU, and Game Tools*, vol. 15, no. 4, pp. 235–251, 2011.

[129] G. Laurent, C. Delalandre, G. de La Rivière, and T. Boubekeur, “Forward light cuts: A scalable approach to real-time global illumination,” in *Computer Graphics Forum*, vol. 35, no. 4. Wiley Online Library, 2016, pp. 79–88.

[130] P. Moreau and P. Clarberg, “Importance sampling of many lights on the gpu,” in *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*, E. Haines and T. Akenine-Möller, Eds. Berkeley, CA: Apress, 2019, pp. 255–283.

[131] O. Olsson, E. Sintorn, V. Kämpe, M. Billeter, and U. Assarsson, “Efficient virtual shadow maps for many lights,” in *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2014, pp. 87–96.

[132] O. Olsson, M. Billeter, E. Sintorn, V. Kämpe, and U. Assarsson, “More efficient virtual shadow maps for many lights,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 6, pp. 701–713, 2015.

[133] T. Harada, J. McKee, and J. C. Yang, “Forward+: A step toward film-style shading in real time,” *GPU Pro*, vol. 4, pp. 115–134, 2013.

[134] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, “Imperfect shadow maps for efficient computation of indirect illumination,” in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 129.

[135] K. Wu, N. Truong, C. Yuksel, and R. Hoetzlein, “Fast fluid simulations with sparse volumes on the GPU,” *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2018)*, vol. 37, no. 2, pp. 157–167, 2018.

[136] M. Gao*, X. Wang*, K. Wu*, A. Pradhana, E. Sifakis, C. Yuksel, and C. Jiang, “Gpu optimization of material point methods,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018)*, pp. 254:1–254:12, 2018, (*Joint First Authors).

[137] C. Dachsbacher and M. Stamminger, “Splatting indirect illumination,” in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 93–100.

[138] H. Dammertz, D. Sewtz, J. Hanika, and H. Lensch, “Edge-avoiding à-trous wavelet transform for fast global illumination filtering,” in *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 2010, pp. 67–75.

[139] A. Keller and W. Heidrich, “Interleaved sampling,” in *Rendering Techniques*. Springer, 2001, pp. 269–276.

[140] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek, “Interactive global illumination,” 2002.

[141] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche, “Non-interleaved deferred

shading of interleaved sample patterns," in *Graphics Hardware*, 2006, pp. 53–60.

[142] P. Hedman, T. Karras, and J. Lehtinen, "Sequential monte carlo instant radiosity," in *Proceedings of the 20th ACM SIGGRAPH symposium on interactive 3D graphics and games*. ACM, 2016, pp. 121–128.

[143] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha, "Fast bvh construction on gpus," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 375–384.

[144] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*. ACM, 1968, pp. 307–314.

[145] F. Zafar, M. Olano, and A. Curtis, "Gpu random numbers via the tiny encryption algorithm," in *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 2010, pp. 133–141.

[146] B. Walter, K. Bala, M. Kulkarni, and K. Pingali, "Fast agglomerative clustering for rendering," in *2008 IEEE Symposium on Interactive Ray Tracing*. IEEE, 2008, pp. 81–86.

[147] C. Wyman, "Exploring and expanding the continuum of oit algorithms," in *High Performance Graphics*, 2016, pp. 1–11.

[148] E. Kilgariff, H. Moreton, N. Stam, and B. Bell, "NVIDIA Turing Architecture In-Depth," https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/, 2018, [Online; accessed 19-January-2021].

[149] B. Bitterli, "Correlations and reuse for fast and accurate physically based light transport," Ph.D. dissertation, Dartmouth College, 2021.

[150] C. Wyman and A. Panteleev, "Rearchitecting Spatiotemporal Resampling for Production," in *High-Performance Graphics - Symposium Papers*, 2021.

[151] N. Benty, K.-H. Yao, P. Clarberg, L. Chen, S. Kallweit, T. Foley, M. Oakes, C. Lavelle, and C. Wyman, "The Falcor rendering framework," 08 2020. [Online]. Available: https://github.com/NVIDIAGameWorks/Falcor

[152] R. K. Hoetzlein, "GVDB: Raytracing Sparse Voxel Database Structures on the GPU," in *High Performance Graphics*, 2016, pp. 109–117.

[153] K. Museth, "Vdb: High-resolution sparse volumes with dynamic topology," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, 2013.

[154] N. Hofmann, J. Hasselgren, P. Clarberg, and J. Munkberg, "Interactive path tracing and reconstruction of sparse volumes," *Proceedings of the ACM in Computer Graphics and Interactive Techniques*, vol. 4, no. 1, Apr. 2021.

[155] H. Halen, A. Brinck, K. Hayward, and X. Bei, "Global illumination based on surfels," in *SIGGRAPH Courses; Advances in Real-Time Rendering*, 2021.

[156] T. Müller, B. Mcwilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance

sampling," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, oct 2019.

[157] D. Lin, C. Wyman, and C. Yuksel, "Fast volume rendering with spatiotemporal reservoir resampling," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2021)*, vol. 40, no. 6, pp. 279:1–279:18, 2021. [Online]. Available: http://doi.acm.org/10.1145/3478513.3480499

[158] K. Nabata, K. Iwasaki, and Y. Dobashi, "Resampling-aware weighting functions for bidirectional path tracing using multiple light sub-paths," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 2, mar 2020.

[159] C. Wyman and A. Panteleev, "Rearchitecting spatiotemporal resampling for production," in *ACM/EG Symposium on High Perfrormance Graphics*, 2021, pp. 23–41.

[160] E. Veach, *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[161] F. Liang and S. Cheon, "Monte carlo dynamically weighted importance sampling for spatial models with intractable normalizing constants," vol. 197, p. 012004, dec 2009. [Online]. Available: https://doi.org/10.1088/1742-6596/197/1/012004

[162] M. Manzi, F. Rousselle, M. Kettunen, J. Lehtinen, and M. Zwicker, "Improved sampling for gradient-domain metropolis light transport," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, pp. 1–12, 2014.

[163] W. Jakob and S. Marschner, "Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–13, 2012.

[164] M. Kettunen, "Gradient-domain methods for realistic image synthesis," Ph.D. dissertation, Aalto University, 2020.

[165] L. Szésci, L. Szirmay-Kalos, and C. Kelemen, "Variance reduction for russian-roulette," 2003.

[166] S. Kallweit, P. Clarberg, C. Kolb, K.-H. Yao, T. Foley, Y. He, L. Wu, L. Chen, T. Akenine-Moller, C. Wyman, C. Crassin, and N. Benty, "The Falcor rendering framework," 12 2021, https://github.com/NVIDIAGameWorks/Falcor. [Online]. Available: https://github.com/NVIDIAGameWorks/Falcor

[167] B. Bitterli, W. Jakob, J. Novák, and W. Jarosz, "Reversible jump metropolis light transport using inverse mappings," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 1, pp. 1–12, 2017.

[168] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces." *Rendering Techniques*, vol. 2007, p. 18th, 2007.

[169] P. S. Heckbert, "Adaptive radiosity textures for bidirectional ray tracing," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, 1990, pp. 145–154.

[170] D. Lin and C. Yuksel, "Real-time rendering with lighting grid hierarchy," *Proceedings of the ACM in Computer Graphics and Interactive Techniques (Proceedings of I3D 2019)*, vol. 2, no. 1, pp. 8:1–8:17, 2019.

[171] ——, "Real-time stochastic lightcuts," *Proceedings of the ACM in Computer Graphics and Interactive Techniques (Proceedings of I3D 2020)*, vol. 3, no. 1, pp. 5:1–5:18, 2020.

[172] Y. Tokuyoshi and T. Harada, "Hierarchical russian roulette for vertex connections," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, jul 2019.

[173] C. R. A. Chaitanya, L. Belcour, T. Hachisuka, S. Premoze, J. Pantaleoni, and D. Nowrouzezahrai, "Matrix bidirectional path tracing," in *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations*, ser. SR '18. Goslar, DEU: Eurographics Association, 2018, p. 23–32. [Online]. Available: https://doi.org/10.2312/sre.20181169

[174] S. Popov, R. Ramamoorthi, F. Durand, and G. Drettakis, "Probabilistic connections for bidirectional path tracing," *Computer Graphics Forum*, vol. 34, no. 4, p. 75–86, jul 2015.

[175] E. P. Lafortune and Y. D. Willems, "Bi-directional path tracing," in *Proc. the International Conference on Computational Graphics and Visualization Techniques*, vol. 93, Dec. 1993, pp. 145–153.

[176] T. Müller, M. Gross, and J. Novák, "Practical path guiding for efficient light-transport simulation," *Computer Graphics Forum*, vol. 36, no. 4, pp. 91–100, 2017.

[177] J. Vorba, O. Karlík, M. Šik, T. Ritschel, and J. Křivánek, "On-line learning of parametric mixture models for light transport simulation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–11, 2014.

[178] E. Veach and L. J. Guibas, "Metropolis light transport," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, USA, 1997, p. 65–76.

[179] C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka, "A simple and robust mutation strategy for the metropolis light transport algorithm," *Computer Graphics Forum*, vol. 21, no. 3, pp. 531–540, 2002.

[180] J. Pantaleoni, "Online path sampling control with progressive spatio-temporal filtering," *SN Computer Science*, no. 279, aug 2020.

[181] J. Zhu, Y. Bai, Z. Xu, S. Bako, E. Velázquez-Armendáriz, L. Wang, P. Sen, M. Hašan, and L.-Q. Yan, "Neural complex luminaires: Representation and rendering," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, jul 2021.

[182] M. Nießner and C. Loop, "Analytic displacement mapping using hardware tessellation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, pp. 1–9, 2013.

[183] D. Lin, K. Shkurko, I. Mallett, and C. Yuksel, "Dual-split trees," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2019, pp. 1–9.

[184] D. Lin, E. Vasiou, C. Yuksel, D. Kopta, and E. Brunvand, "Hardware-accelerated dual-split trees," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 3, no. 2, pp. 1–21, 2020.

[185] L. Seiler, D. Lin, and C. Yuksel, "Automatic gpu data compression and address swizzling for cpus via modified virtual address translation," in *Symposium on Interactive 3D Graphics and Games*, 2020, pp. 1–10.

[186] ——, "Compacted cpu/gpu data compression via modified virtual address translation," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 3, no. 2, pp. 1–18, 2020.

[187] D. Lin, L. Seiler, and C. Yuksel, "Hardware adaptive high-order interpolation for real-time graphics," in *Computer Graphics Forum*, vol. 40, no. 8. Wiley Online Library, 2021, pp. 1–16.

[188] A. Burnes, "Quake ii rtx: Re-engineering a classic with ray tracing effects on vulkan," https://www.nvidia.com/en-us/geforce/news/quake-ii-rtx-ray-tracing-vulkan-vkray-geforce-rtx/, 2019, [Online; accessed 9-March-2022].

[189] R. G. Bartle, *The elements of integration and Lebesgue measure*. John Wiley & Sons, 2014.