A Gentle Introduction to ReSTIR

Path Reuse in Real-Time

Chris Wyman **Markus Kettunen** **Daqi Lin**
Benedikt Bitterli **Cem Yuksel** **Wojciech Jarosz**
Pawel Kozlowski **Giovanni De Francesco**

March 4, 2024

More information: `https://intro-to-restir.cwyman.org`

SIGGRAPH '23 Courses, August 6–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0145-0/23/08.
https://doi.org/10.1145/3587423.3595511

Abstract

In recent years, reservoir-based spatiotemporal importance resampling (ReSTIR) algorithms appeared out of nowhere to take parts of the real-time rendering community by storm, with sample reuse speeding direct lighting from millions of dynamic lights [1], diffuse multi-bounce lighting [2], participating media [3], and even complex global illumination paths [4]. Highly optimized variants (e.g. [5]) can give 100× efficiency improvement over traditional ray- and path-tracing methods; this is key to achieve 30 or 60 Hz framerates. In production engines, tracing even one ray or path per pixel may only be feasible on the highest-end systems, so maximizing image quality per sample is vital.

ReSTIR builds on the math in Talbot et al.'s [6] resampled importance sampling (RIS), which previously was not widely used or taught, leaving many practitioners missing key intuitions and theoretical grounding. A firm grounding is vital, as seemingly obvious "optimizations" arising during ReSTIR engine integration can silently introduce conditional probabilities and dependencies that, left ignored, add uncontrollable bias to the results.

In this course, we plan to:

1. Provide concrete motivation and intuition for why ReSTIR works, where it applies, what assumptions it makes, and the limitations of today's theory and implementations;
2. Gently develop the theory, targeting attendees with basic Monte Carlo sampling experience but without prior knowledge of resampling algorithms (e.g., Talbot et al. [6]);
3. Give explicit algorithmic samples and pseudocode, pointing out easily-encountered pitfalls when implementing ReSTIR;
4. Discuss actual game integrations, highlighting the gotchas, challenges, and corner cases we encountered along the way, and highlighting ReSTIR's practical benefits.

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[2]: Ouyang et al. (2021), 'ReSTIR GI: Path Resampling for Real-Time Path Tracing'

[3]: Lin et al. (2021), 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

[5]: Wyman et al. (2021), 'Rearchitecting Spatiotemporal Resampling for Production'

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

Course Format & Prerequisites

This is 3 hr course of **intermediate** difficulty level.

We assume attendees understand basic ray tracing and calculus. We hope attendees will have seen the rendering equation, Monte Carlo integration, importance sampling, and related statistics, but we will provide a brief review of these concepts as we gently introduce the mathematics of resampling and ReSTIR.

Target Audience

This course targets students, researchers, and rendering engineers interested in the efficiency gains resampling promises for real-time rendering but have not read or closely followed recent papers, have difficulty gaining intuition for the mathematics of resampling, have questions about corner cases, desire to hear about the challenges and benefits of integrating ReSTIR

in production renderers, or wonder what open problems remain that may motivate future research.

Our technical content starts by reviewing basic Monte Carlo integration, but we will not spend significant time on this review. Our audience should have at least passing familiarity with traditional Monte Carlo integration and importance sampling, e.g., from a graduate computer graphics or image synthesis course.

Why a SIGGRAPH Course in 2023?

After our first ReSTIR paper [1], the algorithm looked very compelling but we felt our knowledge of its full abilities and constraints was limited. With subsequent papers expanding our knowledge [2–5, 7, 8] and our experience taking the research to shipping products [9–11], we now feel confident we can present a usable, understandable, and theoretically sound introduction.

Contemporaneous to our work, we have seen a large uptick in ReSTIR Twitter discussions, blog posts on deciphering our theory [12–14], virtual graphics meetups to learn ReSTIR, slides from graduate graphics courses [15], indie and R&D game developer experiments [16, 17], publications from other labs [18–23], plugins for modelling packages [24, 25], and student final project implementations floating around on the web, in addition to the usual smattering of e-mail queries asking for more details.

This variety and quantity of interest suggests a coherent, gentle introduction to resampling theory would be welcomed by the rendering community, help accelerate further research and adoption, and reduce independently duplicated stumbles on the more common pitfalls encountered when using ReSTIR.

Course Syllabus (3 hours)

	Topic	Speaker
5 min	Welcome and introduction	*Wyman*
15 min	Motivation; why consider ReSTIR?	*Yuksel*
20 min	Resampled importance sampling (RIS)	*Kettunen*
15 min	RIS and direct lighting	*Bitterli*
15 min	Spatiotemporal sample reuse and MIS	*Bitterli*
15 min	Reusing samples between domains	*Kettunen*
20 min	Extending sample reuse to paths	*Lin*
15 min	Making ReSTIR fast: sampler optimization	*Lin*
15 min	Making ReSTIR fast: low-level optimization	*Wyman*
25 min	ReSTIR integration in *Cyberpunk 2077*	*Kozlowski* & *De Francesco*
10 min	Open problems and future directions	*All*
10 min	Audience Q & A	*All*

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[2]: Ouyang et al. (2021), 'ReSTIR GI: Path Resampling for Real-Time Path Tracing'

[3]: Lin et al. (2021), 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

[5]: Wyman et al. (2021), 'Rearchitecting Spatiotemporal Resampling for Production'

[7]: Bitterli (2022), 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'

[8]: Boksansky et al. (2021), 'Rendering Many Lights with Grid-Based Reservoirs'

[9]: NVIDIA (2021), *NVIDIA® RTX Direct Illumination*

[10]: Battaglia (2021), *Sword and Fairy 7 is the cutting-edge PC exclusive nobody's talking about*

[11]: Burnes (2022), *Portal with RTX Out Now: A Breathtaking Reimagining Of Valve's Classic With Full Ray Tracing & DLSS 3*

[12]: Cao (2022), *Understanding The Math Behind ReSTIR DI*

[13]: Guertault (2022), *Reading list on ReSTIR*

[14]: Sachdeva (2021), *Spatiotemporal Reservoir Resampling (ReSTIR) - Theory and Basic Implementation*

[15]: Bikker (2023), *Lecture 14 - "TAA & ReSTIR"*

[16]: Stachowiak (2022), *Global Illumination in 'kajiya' Renderer*

[17]: Zyanide (2023), *Shared Twitter results for Jedi Outcast integration*

[18]: Badke (2021), 'Next event estimation via reservoir-based spatio-temporal importance resampling'

[19]: Ciklabakkal et al. (2022), 'Single-Pass Stratified Importance Resampling'

[20]: Boissé (2021), 'World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination'

[21]: Hermans (2022), 'The Effectiveness of the ReSTIR Technique When Ray Tracing a Voxel World'

[22]: Liu et al. (2023), 'Light Subpath Reservoir for Interactive Ray-Traced Global Illumination'

[23]: Ogaki (2021), 'Vectorized Reservoir Sampling'

[24]: Krake (2021), *hdRstr: A ReSTIR/RTXDI-based Hydra Render Delegate*

[25]: Krake (2022), *blRstr: A ReSTIR/RTXDI-based Blender Render Engine*

About the Contributors

All authors discussed and will help prepare course directions, ideas, schedule, and notes. But due to other obligations, not all will actually present at SIGGRAPH. Please see the proposed schedule for details.

Chris Wyman *(Organizer)* is a Distinguished Research Scientist at NVIDIA, in Redmond, WA, overseeing and participating in research improving sample reuse, including helping transfer knowledge to (internal and external) product groups looking to accelerate their renderers. He has spoken in, co-authored, or organized six different SIGGRAPH courses and was an Associate Professor at the University of Iowa for 10 years. Chris has a PhD from the University of Utah and a BS from the University of Minnesota.

Markus Kettunen *(Speaker)* is a Senior Research Scientist at NVIDIA, in Helsinki, and has been vital in evolving the statistical theory behind resampling to remain robust and unbiased even when borrowing samples from different integration domains. Before joining NVIDIA as a postdoc, he received a PhD from Aalto University, a MSc from the University of Helsinki, and has experience working on Weta's Manuka renderer and developing graphics solutions for the automotive industry.

Daqi Lin *(Speaker)* is a Research Scientist at NVIDIA, in Redmond, WA. As a graduate student, he led development and implementation of complex path resampling in volumetric media and for complex non-diffuse paths. At NVIDIA he helped make this code faster and more robust, and is continuing to evolve ReSTIR to better reduce variance and correlations for more complex path types. He has a PhD from the University of Utah and a BS from the National University of Singapore.

Benedikt Bitterli *(Speaker)* is a Senior Research Scientist at NVIDIA, in Redmond, WA, where he has been investigating a variety of rendering problems including sampling, new material models, and applications of deep learning in real-time rendering. He led the research for the first ReSTIR paper targeting direct lighting for dynamic many-light scenes. He has a PhD from Dartmouth College, a BS and MSc from ETH Zurich, and has both research and production experience from Disney Research and Walt Disney Animation Studios.

Cem Yuksel *(Speaker)* is an Associate Professor at the University of Utah in Salt Lake City, UT, a Senior Scientist at Roblox Research, and the Founder of Cyber Radiance LLC. His research covers numerous areas in computer graphics, from hardware to physically-based simulations to geometry and modeling to sampling and rendering. He has co-authored and organized two prior SIGGRAPH courses. Before joining the University of Utah, he did a postdoc at Cornell University and has a PhD from Texas A&M University.

Wojciech Jarosz *(Co-Author)* is an Associate Professor at Dartmouth College, where he co-founded the Visual Computing Lab. Before joining Dartmouth, he was a Senior Research Scientist in charge of rendering at Disney Research Zurich and an adjunct lecturer at ETH Zurich. He has a PhD and MS

from UC San Diego and a BS from the University of Illinois at Urbana-Champaign.

Pawel Kozlowski *(Speaker)* is a Principal Developer Technology Engineer in the Developer and Performance Technology group at NVIDIA with over 10 years of experience in computer graphics. He focuses on integrating path tracing into the current game engines, which spans the domains of sampling and denoising algorithms, and delivering the best GPU performance possible to gamers on the GeForce platform. Unconventionally, he developed his interest in real-time graphics while pursuing his Ph.D. in medical imaging at GE Healthcare and the University of Oslo, Norway, where he worked with volume rendering for 3D cardiac ultrasound.

Giovanni De Francesco *(Speaker)* is a Senior Lighting Technical Artist at CD Projekt Red where he strives for world-class aesthetics in games. He designs and tests tools for the lighting team, ensuring best practices even when using cutting edge rendering techniques. Before joining CD Projekt Red, he worked at inVRsion on lighting in VR, Clay Milano on commercials, in addition to lighting roles elsewhere.

Contents

Abstract i
Course format & Prerequisites i
Why a SIGGRAPH Course in 2023? ii
Course Syllabus ii
About the Contributors iii

Contents **v**

1 Introduction **1**
1.1 Motivation for ReSTIR 2

2 Preliminaries **4**
2.1 Monte Carlo integration 4
2.2 Supports 5
2.3 Multiple Importance Sampling 6
2.4 Unbiased Contribution Weights 7

3 Resampled Importance Sampling **8**
3.1 Resampled Importance Sampling 9
3.2 MIS weights 12
3.3 Example: resampled importance sampling (RIS) between BSDF and NEE 13
3.4 Inputs with unknown probability density functions (PDFs) 13

4 ReSTIR: Spatiotemporal Reservoir Resampling **15**
4.1 Weighted Reservoir Sampling 15
4.2 Spatiotemporal reuse 16
4.3 Example: ReSTIR for direct illumination 17
4.4 History length 19
4.5 Advanced topics 20

5 Reusing Between Domains **22**
5.1 Preliminaries 22
5.1.1 Shift mappings 22
5.1.2 Jacobian determinants 23
5.2 Reusing samples between domains 23
5.3 MIS between domains 24

6 ReSTIR Path Tracing **27**
6.1 The path integral 27
6.2 RIS with a path tracer 27
6.3 Reuse path samples 28
6.4 What is a good shift mapping? 28
6.5 Common shift mappings 29
6.6 An efficient shift mapping for real-time rendering 30
6.7 Volume rendering 34

7 Making reservoir-based spatiotemporal importance resampling (ReSTIR) fast **37**
7.1 Sampler optimization 37
7.1.1 Neighbor rejection as approximate "MIS weights" . 38
7.1.2 Contribution MIS weights 38
7.1.3 Pairwise MIS weights 39
7.1.4 Biased multiple importance sampling (MIS) Weights 41
7.2 Low-level optimization 42
7.2.1 Sample tiling in ReSTIR DI 43
7.2.2 Lighting with many analytic light types 45
7.2.3 Accelerating hybrid shift 45

8 Experiences in game integration **47**

9 Advice for getting started **48**

Bibliography **50**

List of Abbreviations **53**

Symbols **54**

1 Introduction

1.1 Motivation for ReSTIR 2

Real-time path tracing targeting movie-quality results is a challenging goal. Real-time renderers are often limited to 16 ms per frame on one graphics processing unit (GPU), not hours or days on a render farm, as in many offline renderers. In a real-time context, maximizing efficiency is the *primary* goal, and this goal encourages cross-reuse of information between different pixels and across frames. ReSTIR [1], the iterated application of RIS [6], allows unbiased sample reuse from a large number of frames by repeatedly aggregating multiple neighbor samples into one sample of higher quality.

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

A common question from experienced practitioners asks, "how can you even ensure neighboring samples are relevant to your current pixel?" This is a perceptive question. And the answer is, "very, very carefully." In fact, accounting for sample supports and domains is the *key challenge* with ReSTIR (e.g., see Section 3.2).

But with a bit of thought, it is not very surprising that sample reuse works. After all, modern denoisers [26, 27] and upsamplers reuse and filter colors across pixels. That, too, can be seen as sample reuse between integrands with varying domains. Usually post-process denoisers entirely ignore the support issue, which is why they often reduce energy (among other kinds of bias).

[26]: Schied et al. (2017), 'Spatiotemporal Variance-Guided Filtering'

[27]: Schied et al. (2018), 'Gradient Estimation for Real-Time Adaptive Temporal Filtering'

The enormous advantage of RIS and ReSTIR is they (largely) filter, resample, and reuse samples *before* throwing any information away. This allows constructing unbiased algorithms as we have access to intermediate probabilities, distributions, and samples; post-process denoising can only access colors plus a few explicit guide buffers (usually from the G-buffer [28]).

[28]: Saito et al. (1990), 'Comprehensible Rendering of 3-D Shapes'

Hence, one way to understand ReSTIR is: a filtering technique for sampling distributions—aggregating multiple samples into one with a better PDF. If blending colors in a denoiser improves image quality, maybe we could reduce noise by filtering our PDFs?

In fact, path guiding techniques [29] have shown filtering PDFs can help: they learn sample distributions by fitting PDF families to old samples. ReSTIR streamlines this by skipping the learning—the PDF improves by repeated weighted reuse of existing samples from other pixels and frames.

[29]: Vorba et al. (2019), 'Path Guiding in Production'

In any case, the fact ReSTIR works should not be surprising; it resembles many prior, widely used sampling techniques. One of its key contributions is allowing lazy, streaming, GPU-accelerated variants of these algorithms using weighted reservoir sampling [30].

[30]: Chao (1982), 'A General Purpose Unequal Probability Sampling Plan'

1.1 Motivation for ReSTIR

We assume that the reader is familiar with the path integral formulation of light transport [31]: the incident radiance to a pixel is given by the total over all possible paths from all emitters to the sensor:

[31]: Veach (1997), 'Robust Monte Carlo Methods for Light Transport Simulation'

$$I_i = \int_\Omega h_i(x) f(x) \, \mathrm{d}x, \tag{1.1}$$

where Ω contains all paths of all lengths, h_i is the image filter for the pixel, f is the measurement contribution function, and $\mathrm{d}x$ is the product-area measure.

With a box filter assumption (generalizing to more complex filters is straightforward), we can define a per-pixel domain Ω_i that only includes paths that pass through the pixel,

$$I_i = \int_{\Omega_i} f(x) \, \mathrm{d}x \ . \tag{1.2}$$

Path tracers utilize this by randomly sampling paths X from the camera to the scene, letting the paths bounce at interactions, and contributing the radiance $f(X)$ carried by the path, divided by its sampling probability density $p(X)$, giving a Monte Carlo estimator for I_i:

$$\langle I_i \rangle = \frac{f(X)}{p(X)} \approx I_i. \tag{1.3}$$

This estimate $\langle I_i \rangle$ is unbiased, i.e., correct on average (in terms of signed error), but noisy. The more the PDF p deviates from f, the noisier $\langle I_i \rangle$ becomes. A perfect match, where $p \sim f$, gives no noise, i.e., a zero-variance estimate.

This noise can also be decreased by averaging more samples as

$$\langle I_i \rangle = \frac{1}{N} \sum_{j=1}^{N} \frac{f(X_j)}{p(X_j)}, \tag{1.4}$$

but averaging quickly becomes inefficient. Each halving of noise magnitude requires four times the samples. Whenever possible, it is better to make the density p more closely match f. But, this is the fundamental challenge of light transport: predicting the paths that carry a lot of light is *hard*, so matching p to f is *hard*.

This leads to ReSTIR's key premise: while nearby pixels clearly see somewhat different light paths, similar paths still tend to be important for neighbors. Hence, a good path for pixel a, when reused for pixel b with minor modifications, tends to be useful for pixel b. This allows improving pixel b's estimate by reusing *good* samples from pixel a.

But, how do we know what paths are *good*? Good paths carry radiance, i.e., have large f. But we cannot arbitrarily choose to reuse some paths

more without relying on solid mathematical theory, or we will introduce bias. Besides, our goal is producing samples with probability density proportional to f. We do not aim to *just* produce the high-contribution samples.

This is where resampled importance sampling (RIS) helps [6]. Given a sequence of inputs $(X_1, \ldots, X_M)$, RIS calculates weights w_i for all inputs and chooses one sample proportionally to the w_i's such that this selected sample's probability density is (usually) closer to f. RIS aggregates many samples into one sample that is better-distributed. Simplifying a bit, this is why RIS is useful: RIS is an aggregation machine. It takes a number of candidate samples as inputs, and aggregates them into one sample with a better p, decreasing noise.

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

While this one output sample is *not* better than all inputs combined, aggregration has a clear advantage: the output is *just one* sample, not M samples. Processing just one sample is generally cheaper than processing M samples, and this cost difference increases drastically if we *chain* RIS, i.e., resample from RIS-aggregated samples. Assume a RIS resampling from M_2 samples, that each aggregate M_1 samples from the prior frame. The output is an aggregation of $M_1 \times M_2$ samples, while it costs less than $M_1 + M_2$ samples, as borrowing samples tends to be cheaper than generating new ones and the M_1 part was already paid for in the previous frame.

Imagine RIS aggregation iteratively over frames. Every frame, take a new sample for each pixel. Then, combine this new sample with aggregated samples from the previous frame plus samples borrowed from neighboring pixels. While the RIS results are *at best* as good as combining the individual inputs, imagine if we could aggregate (for instance) $1920 \times 1080 \times 10$ samples for each pixel, but at *roughly the cost of one sample* each frame. While such massive aggregation may not be possible in practice, iterated RIS (also known as ReSTIR), often improves image quality equivalent to using hundreds of independent samples per pixel, but at minimal cost.

Preliminaries 2

Before we go into the details of RIS and ReSTIR, we will briefly cover some preliminaries. Readers already familiar with random variables, supports, and Monte Carlo integration may want to skip to 3.

2.1 Monte Carlo integration . . . 4
2.2 Supports . . . 5
2.3 Multiple Importance Sampling . . . 6
2.4 Unbiased Contribution Weights . . . 7

2.1 Monte Carlo integration

Monte Carlo (MC) integration is a technique for numerically approximating integrals. Given an integral of the form

$$I = \int_{\Omega} f(x)\,\mathrm{d}x\ , \tag{2.1}$$

which may be intractable to compute in closed form, Monte Carlo integration approximates I using M randomly-selected samples $X_1, \ldots, X_M$. The function f is evaluated only M times, once for each of these samples.

Notation: We use capital letters X to refer to random variables, in contrast to traditional variables x in Equation 2.1. This will be important when writing equations that include both random and traditional variables, making the math more readable and the potential issues easier to spot.

When using a uniform *distribution* for selecting random samples $X_1, \ldots, X_M$, we can write the Monte Carlo (MC) estimator as

$$I \approx \langle I \rangle = |\Omega| \frac{1}{M} \sum_{i=1}^{M} f(X_i)\ . \tag{2.2}$$

Definition 2.1.1 (Uniform distribution) *In a uniform distribution, every possible sample value has an equal probability (density).*

Thus, the MC estimator in this case is a simple average of $f(X_i)$ values computed at M sample values, scaled by the size of the integration domain $|\Omega|$. Here, $\langle I \rangle$ is an *estimator* of I. It is itself a random variable, which implies that its exact value may not necessarily match the integral. However, provided that the random samples X_i cover the entire integration domain Ω, $\langle I \rangle$ will have the correct *expected value*: $\mathbb{E}[\langle I \rangle] = I$. In other words, it will return the correct answer *on average*. Moreover, this estimator is *consistent*, meaning that as M goes to infinity, $\langle I \rangle$ converges to I.

Definition 2.1.2 (Convergence in probability) *Estimator $\langle I \rangle$ converges* in probability *if, regardless of threshold ε, the probability that $|\langle I \rangle - I| > \varepsilon$ approaches zero for large M.*

It is often advantageous to use a non-uniform distribution of random samples, such that each sample X is associated with a probability of selecting it. The probability *density* of a continuous random variable X is given by its probability density function (PDF), often denoted p, as $p(X)$. With a non-uniform distribution, we can no longer use a simple average and expect to get the correct result. Instead, we must weigh the function evaluations based on their chance of being sampled: locations that have a lower chance of being sampled should get a higher weight to ensure that

$\mathbb{E}[\langle I \rangle] = I$. This leads to a more general form of MC estimator

$$\langle I \rangle = \sum_{i=1}^{M} \frac{1}{M} \frac{f(X_i)}{p_i(X_i)} , \tag{2.3}$$

where the contribution of each sample X_i is divided by its probability density[1] $p_i(X_i)$. Notice that the probability density function p_i can be different for each sample. Though it is commonplace to use the same PDF for all samples, combining samples with different PDFs is a crucial component of ReSTIR.

1: The uniform distribution corresponds to a constant PDF of $p_i(X_i) = 1/|\Omega|$, which reduces Equation 2.3 to Equation 2.2.

Typically, as M increases, $\langle I \rangle$ becomes more likely to be close to I. Yet, it is also possible to use $M = 1$, such that

$$\langle I \rangle = \frac{f(X)}{p(X)}. \tag{2.4}$$

One might expect this to be a highly inaccurate estimator, but, in fact, its accuracy depends on the PDF and how it relates to f. For example, if $f(x)/p(x)$ is a constant value for all $x \in \Omega$, such a p is considered a *perfect* PDF[2] and a single sample is sufficient to perfectly estimate I. Yet, defining a perfect PDF is typically impractical, because it requires knowing the value of I ahead of integration[3]. The goal of ReSTIR is to bring the effective PDF as close as possible to the perfect PDF, such that a small number of samples (such as a single sample) can provide a good MC estimator.

2: A perfect PDF $p(x)$ would differ from $f(x)$ by a constant scale factor.

3: With a perfect PDF, $f(x)/p(x) = I$ for all $x \in \Omega$.

We measure the quality/accuracy of an MC estimator by its *variance*[4], which is the expected (squared) difference of a particular estimation $\langle I \rangle$ from the expected value $\mathbb{E}[\langle I \rangle]$. The lower the variance, the better the accuracy of the MC estimator, which means less noise. Two common approaches for reducing the variance are increasing the sample count M and improving the PDF by making p a better representative of f (times a constant scale factor). ReSTIR provides a mechanism for the latter approach.

4: MC estimator variance can be written as $\mathrm{Var}[\langle I \rangle] = \mathbb{E}[\Big(\langle I \rangle - \mathbb{E}[\langle I \rangle]\Big)^2]$.

Deviation of the expected value $\mathbb{E}[\langle I \rangle]$ from I is called *bias*; an algorithm without bias is called *unbiased*. The MC estimator described above is unbiased under relatively mild conditions we explain below, but breaking the conditions can introduce bias.

2.2 Supports

A *function*'s *support* is simply a fancy name for the part where it is nonzero. For example, the support of the path contribution function f is all paths that carry radiance to the camera. The support of $\max(1 - |x|, 0)$ is $(-1, 1)$. We denote the support of a function f by $\mathrm{supp}(f)$.

Definition 2.2.1 (Function support) *The support of function f, $\mathrm{supp}(f)$, is the set of all x where $f(x) \neq 0$.*

A *random variable*'s support is the values it can take. A uniform random variable X from 0 to 1 has support $\mathrm{supp}(X) = [0, 1]$. A discrete random variable's support is the values it can take with a positive probability. A continuous random variable's support is the values it can take with a positive probability density.

Definition 2.2.2 (Variable support) *The support of random variable X, $\mathrm{supp}(X)$, is the set of all values that X can take.*

The connection between the two concepts with the same name is that if X has PDF p, then $\text{supp}(X) = \text{supp}(p)$, and similarly for discrete random variables and probabilities.

MC integration requires only one condition for unbiasedness: The support of X must contain the support of f, i.e., $\text{supp}(f) \subseteq \text{supp}(X)$; we say X "covers" f for short. It is easy to see why:

$$\mathop{\mathbb{E}}_{X}[\langle I \rangle] = \int_{\text{supp}(X)} \frac{f(x)}{p(x)} \cdot p(x)\,\mathrm{d}x = \int_{\text{supp}(X)} f(x)\,\mathrm{d}x. \tag{2.5}$$

MC integration only occurs over the support of X, and any values of f that lie outside of it will be ignored. Traditionally, this requirement is written as $p(x) > 0$ if $f(x) > 0$; although it is generally well-known, it rarely poses an issue in practice and does not usually factor into algorithm design.

However, ReSTIR-derived techniques differ from the norm and easily run afoul of this requirement. Because ReSTIR mixes samples from many distributions (neighboring or past pixels, etc.) that do not necessarily cover the integrand, we need to take extra care to ensure unbiasedness. We will point these issues out throughout this course, and it is an important issue to keep in mind when designing and implementing ReSTIR-based techniques.

Tip 2.1 It would be confusing to write

$$\mathbb{E}\left[\frac{f(x)}{p(x)}\right] = \int_{\text{supp}(x)} f(x)\,\mathrm{d}x,$$

overloading the same symbol x to mean a specific random variable (x) and an integration variable (x). It is unclear which lowercase x refers to the random variable; hence, our capitalization convention:

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

2.3 Multiple Importance Sampling

MIS [32] is a way of efficiently combining samples from multiple random variables. Consider the naive MC estimator in Equation 2.3. There are two big problems with it: not every random variable is equally good at sampling f everywhere, but in Equation 2.3 we *add* their variances, giving us the worst of all worlds. Simultaneously, we may end up with bias if *any* of the random variables do not cover f.

[32]: Veach et al. (1995), 'Optimally Combining Sampling Techniques for Monte Carlo Rendering'

MIS solves both of these issues by performing a weighted combination instead:

$$\langle I \rangle = \sum_{i=1}^{M} m_i(X_i) \frac{f(X_i)}{p_i(X_i)}, \tag{2.6}$$

where $m_i(X_i)$ is the MIS weight of the ith random variable. In order for Equation 2.6 to be unbiased, the MIS weights must satisfy two conditions:

- $\sum_i m_i(x) = 1$ for any x within the support of f, and
- $m_i(x) = 0$ if $x \notin \text{supp}(X_i)$.

This ensures unbiased results when the *union* of all $\text{supp}(X_i)$ covers f.[5]

5: In fact, the conditions *guarantee* the union of all $\text{supp}(X_i)$ covers f.

The simple average in Equation 2.3 corresponds to $m_i = 1/M$ and is only unbiased if all X_i cover f. Usually, a better choice is the *balance heuristic*

$$m_i(x) = \frac{p_i(x)}{\sum_j p_j(x)}, \tag{2.7}$$

which is an "optimal" weighting scheme in a variance sense[6].

6: That is, if the m_i are assumed positive. Negative weights can do even better [33].

2.4 Unbiased Contribution Weights

So far, we assumed that the PDF $p(x)$ can be evaluated in closed form. However, if generating a sample from X is a complicated process—such as Woodcock tracking [34] or photon mapping [35]—evaluating $p(X)$ may be completely intractable. Luckily, we can still perform unbiased MC integration as long as we know a random variable W_X whose *expected value*, given X, matches the *reciprocal PDF*, $\mathbb{E}[W_X|X] = 1/p(X)$. Then we may use the modified MC estimator[7]

$$\langle I \rangle = f(X) \cdot W_X \quad \text{with} \quad \mathbb{E}[f(X) \cdot W_X] = \mathbb{E}[f(X)/p(X)] = I. \tag{2.8}$$

[34]: Woodcock et al. (1965), 'Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry'

[35]: Jensen (1996), 'Global Illumination Using Photon Maps'

7: With the assumption that X covers f.

It is very surprising that there should be simple formulas allowing such W_X to be evaluated even when $p(X)$ cannot! Yet, multiple instances of this exist in graphics [36, 37] and many more in other fields.

Samples produced with RIS also fall into this category: $p(X)$ is completely intractable (a high-dimensional integral, dimensionality growing at each resampling!) but a corresponding W_X *exists*, with a very cheap formula [1, 6]. In the context of RIS, we term W_X an *unbiased contribution weight* [4], and it is key to generalized reuse across domains, as we will soon see.

[36]: Qin et al. (2015), 'Unbiased Photon Gathering for Light Transport Simulation'

[37]: Zeltner et al. (2020), 'Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints'

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

Resampled Importance Sampling 3

3.1 Resampled Importance Sampling 9
3.2 MIS weights 12
3.3 Example: RIS between BSDF and NEE 13
3.4 Inputs with unknown PDFs 13

The effectiveness of importance sampling depends on the PDF used for generating the samples. However, we often do not have an explicit formulation for the ideal PDF and, even when we do, it may be difficult/impossible to generate random samples with the exact PDF we want.

Resampled importance sampling (RIS) provides a solution for these problems. It takes as input a sequence of candidate samples $(X_1, \dots, X_M)$, gives each candidate a *resampling weight* w_i, picks one of the X_i at random, proportionally to the weights w_i, and outputs the selected sample. This process is equivalent to generating samples with a PDF that can be different than the one used for generating the candidate samples. We can control the resulting PDF based on how we assign the resampling weights.

Candidates are continuous random variables, and the output is a continuous random variable—hence, despite the discrete selection, RIS can be compared to path guiding: it is given random variables ("samples"), and it outputs a continuous random variable with a different distribution. In contrast to traditional path guiding, however, RIS does not learn a distribution based on existing samples, but reuses one (or more) of them at random, effectively aggregating multiple samples into one with a better probability density, sometimes compared to *filtering the distributions*. Similar to path guiding, the output is a continuous random variable with an improved distribution, a PDF p that matches f better and thus produces less variance.

But of course, there is a catch.

The PDF of the sample produced by RIS is typically intractable and cannot be evaluated in real-time. Evaluating the PDF is at least as hard as shading the pixel. So why do RIS and ReSTIR work? Why do we talk about them, if the PDFs cannot be evaluated? Didn't Monte Carlo integration require the p so that we can evaluate $f(x)/p(x)$?

Let's think—what's the role of $1/p(X)$ in the $f(X)/p(X)$ estimator? It's a weight for the sample $f(X)$. Is this weight needed? *Yes, absolutely.* Does the weight *need* to be a PDF? *Not exactly*. What? Well, you see, RIS provides the sample X a weight, which we denote W_X. This weight produces an unbiased contribution $f(X)W_X$ that estimates the integral of f. Weights are needed, but they need not be PDFs.

Definition 3.0.1 (Unbiased Contribution Weight) *An unbiased contribution weight W_X is a random variable such that regardless of f:*

$$\mathbb{E}\left[f(X)W_X\right] = \int_\Omega f(x)\,\mathrm{d}x.$$

This W_X replaces the weight $1/p(X)$, but a single sample X can have many different valid W_X, depending on *which* candidate samples $(X_1, \dots, X_M)$ were used to select X. In other words, W_X is not a deterministic function of X. It is a random variable. These weights W_X replace the $1/p(X)$ factor, and they are ubiquitous in RIS and ReSTIR theory. Hence, we should give them a name. What could be a name for a weight that produces an unbiased contribution? How about *unbiased contribution weight*?

Tip 3.1 We write W_X instead of $W(X)$: this is not a function since it cannot be evaluated at arbitrary X. Do not use in MIS weights!

The original exposition of resampled importance sampling [6], as well as

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

early ReSTIR papers (before the generalized theory [4]), used formulas

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

$$w_i = \frac{\hat{p}(X_i)}{p(X_i)} \quad \text{and} \quad W_X = \frac{1}{\hat{p}(X)} \left(\frac{1}{M} \sum_{i=1}^{M} w_i \right), \tag{3.1}$$

to choose the result X from the X_i proportionally to resampling weights w_i. Above, $\hat{p}(x)$ is the *target function* that the PDF of X approximates better and better with more and more candidates. The *unbiased contribution weight* for the chosen X is W_X, and it replaces $1/p(X)$. Even though the PDF is intractable, it certainly exists and, if everything is done according to the theory, converges to being proportional to $\hat{p}$ when we add more samples.

We follow the generalized formulation [4] that moves the $1/M$ factor into the resampling weights w_i. This is equivalent, but gives simpler math and algorithms. As we will see, the job of the $1/M$ is a *resampling MIS weight*. It is *not* there to average of the w_i weights, as suggested by Equation 3.1. The exposition of RIS that we lay out is thus more akin to

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

$$w_i = \frac{1}{M} \frac{\hat{p}(X_i)}{p(X_i)} \quad \text{and} \quad W_X = \frac{1}{\hat{p}(X)} \sum_{i=1}^{M} w_i, \tag{3.2}$$

where the $1/M$ weight will later turn into a resampling MIS weight m_i, and the variance of the sum $\sum_{i=1}^{M} w_i$ now has an intimate connection to convergence: when the variance of $\sum_{i=1}^{M} w_i$ approaches zero [4], the output PDF approaches the *target PDF*, $\bar{p} = \hat{p} / \int \hat{p}$. The contribution weight W_X also approaches $1/\bar{p}(X)$. Choosing $\hat{p} = f$ turns RIS into a zero-variance estimator in the limit, producing samples proportionally to f.

The target function $\hat{p}$ is also sometimes (inaccurately) called the target PDF, but this misnomer should be avoided; $\hat{p}$ is an unnormalized function, often just the integrand, $\hat{p} = f$, or at least close to it. The $\hat{p}$ however *defines* the target PDF $\bar{p}$; with more and more candidate samples, p approaches $\bar{p}$.

Tip 3.2 Factor $\frac{1}{M}$ cancels out in

$$\Pr[\text{choose } i] = \frac{w_i}{\sum_{j=1}^{M} w_j}.$$

Tip 3.3 The sum $\sum_{i=1}^{M} w_i$ estimates $\hat{p}$'s normalization: By Equation 3.2,

$$\sum_{i=1}^{M} w_i = \hat{p}(X) W_X$$

and with Definition 3.0.1,

$$\mathbb{E}\left[\sum_{i=1}^{M} w_i \right] = \mathbb{E}\left[\hat{p}(X) W_X \right] = \int_\Omega \hat{p}(x) \, \mathrm{d}x.$$

3.1 Resampled Importance Sampling

We define resampled importance sampling by the following process:

1. Take candidates $(X_1, \ldots, X_M)$ in a common domain Ω.
2. Evaluate resampling MIS weights $m_i(X_i)$ for all X_i.
3. Evaluate resampling weights $w_i = m_i(X_i)\, \hat{p}(X_i) W_{X_i}$ for all X_i.
4. Choose X randomly from the X_i proportionally to w_i.
5. Evaluate the unbiased contribution weight $W_X = \frac{1}{\hat{p}(X)} \sum_{j=1}^{M} w_j$.

This process gives us a sample X drawn from a PDF that is approximately proportional to the target function $\hat{p}$; increasingly so with more candidate samples. While X is *one sample*, it in a sense represents many; this representation is encoded into its improved PDF, and reflected in the unbiased contribution weight W_X, as the PDF cannot in practice be evaluated. The returned sample X can be *at best* as good for integration as the candidates

Domain: (intuitively) where an object "lives": real numbers live in $\mathbb{R}$, path vertices live on the scene surfaces. A generic domain is often denoted Ω.

$m_i(X_i)$: the resampling MIS weight of X_i. If all X_i are identically distributed, use $m_i = 1/M$.

W_{X_i}: unbiased contribution weight of X_i. If X_i has a known PDF $p(X_i)$, use $W_{X_i} = \frac{1}{p(X_i)}$.

$\Pr[\text{choose } i] = \frac{w_i}{\sum_{j=1}^{M} w_j}$.

Treat W_X as $1/p(X)$ but note it is an unbiased estimate, not a function of X.

Algorithm 1: Resampled importance sampling.

Input : M: number of candidates to generate.
Output: Sample Y and its unbiased contribution weight W_Y

function randomIndex($w_1, \ldots, w_M$) `// Choose s proportionally to the` w_i
  $r \leftarrow \text{rand}()$
  for $s \leftarrow 1$ **to** M **do**
    if $w_s > 0$ **then**
      $r \leftarrow r - w_s / \sum_i w_i$
      if $r \leq 0$ **then**
        return s
  return $\emptyset$

function ResampledImportanceSampling(M)
  `// Generate candidates (`$X_1, \ldots, X_M$`)`
  for $i \leftarrow 1$ **to** M **do**
    generate X_i
    $w_i \leftarrow m_i(X_i)\, \hat{p}(X_i) W_{X_i}$
  `// Select` Y `from the candidates`
  $Y, W_Y \leftarrow \emptyset, 0$
  $s \leftarrow \text{randomIndex}(w_1, \ldots, w_M)$
  if $s \neq \emptyset$ **then**
    $Y \leftarrow X_s$
    $W_Y \leftarrow \frac{1}{\hat{p}(Y)} \sum_i w_i$
  return Y, W_Y

$X_1, \ldots, X_M$ combined (with the resampling MIS weights m_i). The unbiased contribution weight W_X works as the unknown $1/p(X)$ in Monte Carlo integration. In fact, on average[1] W_X is $1/p(X)$.

To use the returned X, we must know what values it may take, i.e., its support. This support is the union of the input supports, with x where $\hat{p}(x) = 0$ removed[2]. To integrate f using X, the target function $\hat{p}$ must be positive whenever f is non-zero, and the inputs must together cover the support of f. If $\hat{p}(x) = 0$, x is never selected by RIS. To avoid biasing our estimate, we must be able to select samples across all of f's support.

Given such a support, this allows unbiased integration without knowing PDFs. Assuming the inputs together cover f's support with positive $\hat{p}$, then so does the output, and the estimator

$$\langle I \rangle = f(X) W_X \tag{3.4}$$

is an unbiased estimate of the integral of f:[3]

$$\mathbb{E}\left[\langle I \rangle\right] = \int_\Omega f(x)\, \mathrm{d}x = I. \tag{3.6}$$

We show pseudo-code of basic RIS in Algorithm 1.

No samples produced? If all w_i are zero and no sample can be chosen, return a *null sample*, $\emptyset$, with $W_\emptyset = 0$. As usual, do not replace the null

1: Formally, as conditional expectation,

$$\mathbb{E}\left[W_X | X\right] = \frac{1}{p(X)}. \tag{3.3}$$

2: Since most MIS weights have $m_i > 0$ when $\hat{p} > 0$, implying $w_i > 0$, $\text{supp}(X)$ contains all x with $\hat{p}(x) > 0$ that can be generated by some of the inputs: $\text{supp}(X) = \left(\bigcup_{i=1}^{M} \text{supp}(X_i)\right) \cap \text{supp}(\hat{p})$.

3: The general form is (see Equation 2.8)

$$\mathbb{E}\left[f(X) W_X\right] = \int_{\text{supp}(X)} f(x)\, \mathrm{d}x, \tag{3.5}$$

which again shows $\text{supp}(X)$ must cover $\text{supp}(f)$.

sample by immediately drawing another, as that causes bias—it is what it is.[4] If passing a random variable with null sample realization to RIS, include it in other samples' MIS weights as usual: a null sample *realization* does not invalidate the *distribution*.

4: Other than $f(\emptyset) = \hat{p}(\emptyset) = 0$, the null sample gets no special treatment.

Example 3.1.1 (Simple integration) Let us first study a simple case with independent and identically distributed (iid) samples $(X_1, \dots, X_M)$ with known PDF p that covers f, and $\hat{p} > 0$ in f's support. Since we know the input PDFs, we set $W_{X_i} = 1/p(X_i)$, and since the samples are identically distributed, we use constant MIS weights[5]: $m_i(x) = 1/M$.

5: The choice $m_i(x) = \frac{1}{M}$ can only be used if all samples individually cover $\operatorname{supp}(f)$, which is the case here.

We then evaluate the resampling weights

$$\begin{aligned} w_i &= m_i(X_i)\, \hat{p}(X_i) W_{X_i} \\ &= \frac{1}{M} \frac{\hat{p}(X_i)}{p(X_i)}, \end{aligned}$$

and choose index s proportionally to the w_i. We then set $X = X_s$ and evaluate

$$W_X = \frac{1}{\hat{p}(X)} \sum_{j=1}^{M} w_j.$$

Could we use the chosen sample's contribution weight, $W_X = W_{X_s}$? That would lead to bias! The sampling process must be respected.

Since the support of X now covers f, $\langle I \rangle = f(X) W_X$ is an unbiased estimate of the integral of f, i.e.,

$$\mathbb{E}\left[f(X) W_X\right] = \int_\Omega f(x)\, \mathrm{d}x.$$

Since W_X is an unbiased contribution weight, Equation 2.5 says

$$\mathbb{E}\left[f(X) W_X\right] = \int_{\operatorname{supp}(X)} f(x)\, \mathrm{d}x.$$

Since $\operatorname{supp}(X)$ covers $\operatorname{supp}(f)$, we have the result.

In this example, we took M iid candidates, selected one of them, and integrated f with the result. This can be done, but if all we did was integrate, simply averaging the M individual contributions would have been just as good. But, we also got a sample X that aggregates the others in its PDF. In the next example, we will use this to improve direction-sampling in a path tracer.

Example 3.1.2 (BSDF importance sampling) Let us repeat the previous steps, but forget about the function f. Our task is simply to provide a sample with PDF approximately proportional to $\hat{p}$. We repeat the above steps: evaluate m_i, w_i, choose X and evaluate W_X. Assuming each input sample covers $\hat{p}$, we know that X covers exactly $\hat{p}$'s support.

Let us then pretend the X_i are directions, importance sampled with a PDF p. Our $\hat{p}$ is, say, a cheaper proxy for the full BSDF f [6] with the same support. We select one of the M candidates proportionally to the w_i. This results in a sample X with improved distribution, PDF approximately proportional to the proxy $\hat{p}$. We trace a ray in that direction and continue path tracing. Our Monte Carlo estimator uses W_X in place of $1/p(X)$ for the sampled direction.

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

This method of improving the sample distribution can sometimes be

useful with the right parameters, but a lot more can be done if we allow mixing samples from different distributions, such as reusing from different pixels or mixing BSDF-guided samples with light sampling. Since the different samples cover different parts of the integration domain, we need MIS weights.

3.2 MIS weights

While we began our introduction by talking about integrating a function f, the direct purpose of RIS is to produce samples approximately proportionally to $\hat{p}$. It is only then that we worry about integration: if the union of the candidates X_i covers $\text{supp}(\hat{p})$, and $\text{supp}(\hat{p})$ covers $\text{supp}(f)$, then X joined with UCW W_X integrates f without bias.

Earlier, we emphasized that in RIS and ReSTIR, *supports matter*. If the candidate samples have different PDFs, such as when we reuse across pixels, or we resample from candidates generated with a light sampler and candidates generated with a BSDF importance sampler, we need more advanced MIS weights than $1/M$.

Tip 3.4 Rule of thumb: Use $1/M$ weights if and only if all inputs are identically distributed.

If all inputs individually cover the support of the target function, $1/M$ MIS weights are technically unbiased. They could still result in terrible outliers in areas hard for *even one* of the inputs. If *even one* input has zero PDF *anywhere* where $\hat{p} \neq 0$, the $1/M$ weights result in a biased W_X.

When the input samples' PDFs are known, we can replace the $1/M$ weights with the *balance heuristic* [32], removing the bias:

[32]: Veach et al. (1995), 'Optimally Combining Sampling Techniques for Monte Carlo Rendering'

$$m_i(x) = \frac{p_i(x)}{\sum_{j=1}^{M} p_j(x)}. \tag{3.7}$$

Tip 3.5 A MIS weight is a function of one x. A weight that mixes PDFs or other functions at different input realizations x_j is most likely wrong.

The balance heuristic evaluates the probability density of the given x in all the input distributions. While the MIS weight cares about the distributions of the other inputs, it *does not care* about their realized values. An implementation that includes the other realizations in a sample's MIS weight is most likely wrong.

The Achilles' heel of the balance heuristic is that it becomes expensive with large sample counts: the MIS weight for each of the M samples requires evaluating M PDFs, giving a $O(M^2)$ time complexity. This is not a problem for small sample counts, but large sample counts may benefit from more advanced MIS weights such as the *pairwise MIS* [4, 7] (see Section 7.1.3 for more discussion).

Tip 3.6 Keep things simple during implementation. Only replace the balance heuristic with an advanced variant once everything works and it is time for performance!

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'
[7]: Bitterli (2022), 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'

With correctly-computed MIS weights, the supports of individual distributions no longer must *all* cover the support of $\hat{p}$. Instead their *union* must cover $\hat{p}$. In practice, this can be tricky to guarantee unless we have at least one candidate X_i designed to directly target $\hat{p}$: a sample that covers all of $\hat{p}$'s support. We call such a sample *canonical* (Definition 3.2.1), understanding that advanced contexts may add more requirements. This enforces $\text{supp}(X) = \text{supp}(\hat{p})$ for the RIS output, allowing unbiased integration within the support of $\hat{p}$.

Definition 3.2.1 (Canonical sample, simple case) *An input X_i to RIS is called* canonical *if it covers $\hat{p}$ (i.e., $\text{supp}\,\hat{p} \subset \text{supp}\,X_i$).*

In other words, by including a canonical sample covering $\operatorname{supp}(\hat{p})$, or otherwise covering $\operatorname{supp}(\hat{p})$ with the candidates, the RIS output X integrates without bias any function within the support of $\hat{p}$:

$$\mathbb{E}\left[f(X)W_X\right] = \int_{\operatorname{supp}(\hat{p})} f(x)\,\mathrm{d}x. \tag{3.8}$$

3.3 Example: RIS between BSDF and NEE

Assume we want to produce a sample for direct illumination that we can reuse. We draw M_1 candidates from a BSDF importance sampler with PDF p_1, and M_2 candidates from a light sampler with PDF p_2. The PDFs must be converted to the same measure.

Tip 3.7 Area PDFs can be converted to solid angle by multiplying by the geometry term, and solid angle PDFs to area PDFs by dividing by it.

The balance heuristic [32] for the BSDF samples is then

$$m_i(x) = \frac{p_1(x)}{M_1 p_1(x) + M_2 p_2(x)}. \tag{3.9}$$

We evaluate the resampling weights

$$\begin{aligned} w_i &= m_i(X_i)\,\hat{p}(X_i)W_{X_i} \\ &= \left(\frac{p_1(X_i)}{M_1 p_1(x) + M_2 p_2(x)}\right)\hat{p}(X_i)\frac{1}{p_1(X_i)}, \end{aligned}$$

Tip 3.8 Start simple during implementation. Use the full integrand f as $\hat{p}$ and explicitly test and include visibility terms. Only optimize for performance once everything works!

and similarly for the light samples but using p_2 in the numerator in m_i and in the denominator in W_{X_i}. We recommend initially using $\hat{p} = f$, the full path contribution, for ease of implementation, and only testing more performant alternatives after validating everything works with $\hat{p} = f$.

Tip 3.9 Do you love difficult, very frustrating debugging? Implement all performance optimizations simultaneously, rather than starting with a baseline implementation.

Next, we choose index s proportionally to the w_i, set $X = X_s$, and

$$W_X = \frac{1}{\hat{p}(X)} \sum_{j=1}^{M_1+M_2} w_j.$$

This unbiased contribution weight replaces the intractable $1/p(X)$ factor. We now have a better-distributed sample X for direct illumination, covering the full support of f, which is ready to share with other pixels.

Technically, using $1/(M_1 + M_2)$ MIS weights does not result in bias, since both PDFs cover all contributing direct illumination. However, the noise level would be as if evaluating direct illumination with only BSDF sampling—terrible. MIS weights in RIS are just as important as in traditional Monte Carlo integration.

Tip 3.10 If $\hat{p} = f$ (the ideal case), the contribution of one RIS sample X is

$$f(X)W_X = \sum_{i=1}^{M} m_i(X_i)\, f(X_i)W_{X_i} \tag{3.10}$$

which does not depend on the chosen candidate, and equals the Monte Carlo estimate with the same candidates. (When $\hat{p}$ is not proportional to f, the estimate may have more noise.)

3.4 Inputs with unknown PDFs

As before, assume a sequence of inputs $(X_1, \ldots, X_M)$ with varying distributions. Assume the inputs are sampled with RIS, and we only know unbiased contribution weights $W_{X_1}, \ldots, W_{X_M}$.

We already know how to use RIS with unbiased contribution weights. The challenge is the MIS weights: we cannot use anything that requires knowing the PDFs.

As input samples are generated with RIS, they are distributed approximately proportionally to the target functions $\hat{p}_i$ used for resampling; we assume each input is associated with a target function $\hat{p}_i$ that we will use as a proxy for the unknown PDFs p_i.

This results in the following, generalized balance heuristic [4]:

$$m_i(x) = \frac{\hat{p}_i(x)}{\sum_{j=1}^{M} \hat{p}_j(x)}. \tag{3.11}$$

At each iteration, we guarantee that the random variables' supports exactly match that of their target function[6]; this guarantees unbiasedness. We do this by adding a canonical sample if the candidates X_i would not otherwise cover the target function $\hat{p}$.

We form a canonical sample X_c by RIS from one or more iid samples with PDF covering $\text{supp}(\hat{p})$. Multiple candidates are often recommended for a good distribution, but a single candidate can be used as well.

With these MIS weights, we are well equipped for reusing samples between pixels and frames, but only within the same domain and without modification at reuse.

Tip 3.11 Samples coming from RIS have the same supports as their target functions, $\text{supp}(p_i) = \text{supp}(X_i) = \text{supp}(\hat{p}_i)$, and generally approximate the target PDF better with more inputs, especially after iterative resampling.

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

Tip 3.12 Since the $\int \hat{p}_i$ are not normalized, this could distort MIS weights if the $\int \hat{p}_i$ have significant variation across pixels.

6: I.e., $\hat{p} = 0$ exactly when $p = 0$.

Definition 3.4.1 (Canonical sample, $\hat{p}_i$-MIS) *An input X_i to RIS is called* canonical *if it uses $\hat{p}_i = \hat{p}$ and covers $\hat{p}$ (i.e., $\text{supp}\,\hat{p} \subset \text{supp}\,X_i$).*

Tip 3.13 A sample X_i that does not come from RIS can be used with $\hat{p}_i$-based MIS weights, if X_i

- is given a target function $\hat{p}_i$
- covers $\text{supp}(\hat{p}_i)$
- is replaced with a null sample $\emptyset$ if $\hat{p}_i = 0$.

This guarantees $\text{supp}(X_i) = \text{supp}(\hat{p})$, making $\hat{p}_i$-based MIS weights unbiased, even if not always ideal.

Tip 3.14 MIS weights of null samples always zero, $m_i(\emptyset) = 0$.

ReSTIR: Spatiotemporal Reservoir Resampling

4

The RIS algorithm is an effective way of improving the distribution of samples. However, for complex target functions $\hat{p}$ and poorly distributed initial candidates, the number M of candidates required for good sampling might far exceed the computational budget. Spatiotemporal Reservoir Resampling (ReSTIR) addresses this issue by chaining invocations of RIS and reusing samples spatially and temporally. We begin this chapter by introducing *reservoir resampling*, a practical improvement on RIS, before describing spatiotemporal reuse.

4.1 Weighted Reservoir Sampling 15
4.2 Spatiotemporal reuse 16
4.3 Example: ReSTIR for direct illumination 17
4.4 History length 19
4.5 Advanced topics 20

4.1 Weighted Reservoir Sampling

In order to select the output sample, the RIS algorithm in Algorithm 1 needs to generate and store all candidates[1] up-front before selecting the output sample in a second pass. This can be a nuisance in practice, especially on parallel systems such as GPUs.

1: ReSTIR literature often also calls the RIS inputs *candidate samples*. Both terms are correct.

Weighted reservoir sampling (WRS) [30] is a family of algorithms for sampling one (or more) elements from a (weighted) stream of samples in a single pass over the data without storing it, and is a perfect fit for RIS. Weighted reservoir sampling (WRS) processes the elements of the input stream in order, maintaining a reservoir of the currently selected sample. At any point in the stream, WRS possibly replaces the sample in the reservoir with the next sample in the stream. It does so with a with probability such that the sample in the reservoir is drawn from the desired distribution over all elements processed thus far. When the stream ends, the reservoir is returned[2].

[30]: Chao (1982), 'A General Purpose Unequal Probability Sampling Plan'

2: Evaluating MIS weights $m_i(X_i)$ may require knowing the distributions of the other inputs, but not their realizations.

WRS comes in many flavors and can be extended to maintain multiple samples in the reservoir. We refer to Chao [30] and Bitterli et al. [1] for details. We give pseudo-code of RIS implemented with WRS in Algorithm 2, the combination of which we call reservoir resampling.

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

Algorithm 2: Reservoir Resampling.

```
class Reservoir
    Y, W_Y ← ∅, 0                                   // The output sample
    w_sum ← 0                                       // The sum of weights
    function update(X_i, w_i)
        w_sum ← w_sum + w_i
        if rand() < (w_i / w_sum) then
            Y ← X_i
function Resample(M)
    Reservoir r
    for i ← 1 to M do
        generate X_i
        w_i ← m_i(X_i) p̂(X_i) W_{X_i}
        r.update(X_i, w_i)
    if r.Y ≠ ∅ then
        r.W_Y ← (1 / p̂(r.Y)) r.w_sum
    return r
```

4.2 Spatiotemporal reuse

The idea of spatiotemporal reuse is simple: Say we render an image, and use RIS at each pixel to produce a sample. We can only invest in M candidates for each pixel, which limits the sample quality we can obtain. However, the integrands (and target distributions) of pixels within a small neighborhood are likely to be similar. Hence, for a given pixel, the samples produced by RIS at its neighbors are great reuse candidates. This immediately inspires spatial and temporal reuse, leading to the following reuse pattern:

Initial candidates We produce a sample, approximately distributed proportionally to $\hat{p}$ by RIS, from one or more independent samples. If the inputs have identical distributions, we may use $m_i = 1/M$.

Spatial reuse After producing a new sample for each pixel with RIS, each pixel identifies a set of spatial neighbors (e.g., picked randomly from a disk) and invokes RIS again, resampling from its own sample and the sample of each selected neighbor. This process can be repeated multiple times, repeatedly improving the distributions, at the cost of increased correlation. Since the distributions of the inputs vary, advanced MIS weights like the generalized balance heuristic are required.

Tip 4.1 Do *not* choose which spatial neighbors to reuse based on the random samples stored at these neighbors, as this causes bias!

Temporal reuse Reuse can be extended in time as well. In an animation, a pixel's ideal distribution across two adjacent frames is often similar. This allows combining samples of prior frames and the current frame using RIS, after matching pixels with appropriate motion vectors. If temporal reuse occurs each frame, samples feed forward through time indefinitely, continually improving the sampling distribution. If temporal reuse is

followed by spatial reuse, samples from prior frames can also spread spatially, leading to very rapid spread of good samples. Temporal reuse also requires advanced MIS weights; access to previous frames' target functions is required to remove all bias.

One natural order of these steps, per frame, is initial candidate generation followed by temporal reuse, followed by spatial reuse, followed by integration with the selected sample as $\langle I \rangle = f(X)W_X$.

4.3 Example: ReSTIR for direct illumination

Now, let's apply RIS and ReSTIR to direct lighting.

Direct lighting encompasses the contributions of all length-3 paths, i.e., light paths that originate from a light source, are reflected off a surface or particle, and directly reach the sensor. Indexing from the sensor, direct illumination thus consists of paths $[\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2]$, where $\mathbf{x}_0$ is on the image plane, $\mathbf{x}_1$ is the primary hit, and $\mathbf{x}_2$ lies on a emissive light surface. See Figure 4.1 for an illustration. Define the set of points on emissive surfaces as A; then, $\mathbf{x}_2 \in A$.

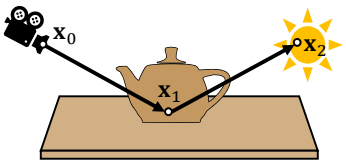

Figure 4.1: An example direct lighting path.

Vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ may be deterministic or depend on randomized lens coordinates. Post-randomization, we treat their values as fixed, making $\mathbf{x}_2$ the only free variable. In this context, our paths are functions of only $\mathbf{x}_2$, expressed in the tuple form:[3]

$$\bar{x} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2] \,. \tag{4.1}$$

3: We gray $\mathbf{x}_0$ and $\mathbf{x}_1$ to emphasize the path is now a function of only $\mathbf{x}_2$.

The remaining challenge is to integrate $\mathbf{x}_2$ over the surface geometry. We plan to improve the distribution of $\mathbf{x}_2$ by spatiotemporal sharing between pixels and frames. We treat $\mathbf{x}_0$ and $\mathbf{x}_1$ as constants, possibly randomized independently for each pixel and frame.

Assuming we have chosen $\mathbf{x}_0$ and $\mathbf{x}_1$, we still need to estimate $L(\mathbf{x}_1 \to \mathbf{x}_0)$ to finish the pixel color estimate (Equation 1.2). We need to integrate

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f_s(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) V(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) L_e(\mathbf{x}_2 \to \mathbf{x}_1) \, \mathrm{d}\mathbf{x}_2, \tag{4.2}$$

where f_s is the BSDF at $\mathbf{x}_1$, L_e is the emission from $\mathbf{x}_2$ in the $\overline{\mathbf{x}_2\mathbf{x}_1}$ direction, G is the geometry term, and V is the visibility term. Treating $\mathbf{x}_0$ and $\mathbf{x}_1$ as constants, the integrand is a function of only $\mathbf{x}_2$,

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_2) \, \mathrm{d}\mathbf{x}_2. \tag{4.3}$$

The vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ vary by pixel index i, giving us a pixel-dependent integrands f_i over the same space A. This defines our ReSTIR context: We aim to improve the distribution of our direct illumination paths by sharing vertices $\mathbf{x}_2$ between pixels. To do so, we resample $\mathbf{x}_2$ from one or more independent *canonical* samples covering the current pixel, plus samples borrowed from other pixels and frames.

Tip 4.2 A *canonical* sample fully covers the support of the target function, while samples reused from neighbors might not, e.g., due to differences in visibility.

In summary, our goal is to leverage resampling to cheaply generate or borrow multiple candidate samples for $\mathbf{x}_2$ such that it obtains an accurate distribution that gives low integration variance.

Next, we outline how to perform the sampling. All resampling steps require defining the target functions, so we start with it.

Target function To define the target function, we build on the integrand in Equation 4.2. For simplicity of notation, we replace $\mathbf{x}_2$ with x and treat $\mathbf{x}_0$ and $\mathbf{x}_1$ as implicit constants. The integrand for direct illumination (with unspecified pixel index) is $f(x) = f_s(x)G(x)V(x)L_e(x)$, and we recommend starting with the same target function, $\hat{p} = f$:

$$\hat{p}(x) = f_s(x)G(x)V(x)L_e(x). \tag{4.4}$$

Talbot et al. [6] drops the visibility term from the target function $\hat{p}$ for performance reasons, using $\hat{p}(x) = f_s(x)G(x)L_e(x)$. This worsens the theoretical limit distribution and requires guaranteeing additional conditions to remain correct, but may lead to better efficiency in practice.

Tip 4.3 We *strongly* recommend first including visibility in your target function. Correcting an optimized method is exponentially harder than retaining correctness while optimizing.

[6]: Talbot et al. (2005), 'Importance Resampling for Global Illumination'

Initial candidates ReSTIR starts by generating canonical samples for each pixel, e.g., with RIS from multiple canonical inputs. In the case of direct lighting, we may pick some number M samples on emitting surfaces with a standard light sampler, and pick one with RIS, with $1/M$ MIS weights. We present alternatives in Section 4.5, but again recommend first finishing the simplest possible correct base implementation.

After initial candidate generation, we perform spatial and temporal reuse.

This is relatively straightforward. We propose starting with the generalized balance heuristic MIS weights, using target functions $\hat{p}_j(x)$, depending on pixel j's sensor and primary vertices $\mathbf{x}_{j,0}$ and $\mathbf{x}_{j,1}$, and x in place of $\mathbf{x}_2$. The MIS weights are then

$$m_i(x) = \frac{\hat{p}_i(x)}{\sum_{j=1}^{M} \hat{p}_j(x)}. \tag{4.5}$$

For example, with $\hat{p} = f$, we have

$$m_i(x) = \frac{f(x \to \mathbf{x}_{i,1} \to \mathbf{x}_{i,0})}{\sum_{j=1}^{M} f(x \to \mathbf{x}_{j,1} \to \mathbf{x}_{j,0})}, \tag{4.6}$$

with $f = f_s \cdot G \cdot V \cdot L_e$.

Spatial reuse For spatial reuse, we recommend picking a suitable number of pixels from the relative vicinity of the current pixel, e.g., a square or a disk[4]. Looking at G-buffer values to heuristically choose similar pixels should be fine, as long as the decisions are *not* based on the *samples* stored in the reservoirs. Using $1/M$ weights generally leads to non-convergence to $\hat{p}$ distribution, and, bias[5].

4: Section 7.1.1 discusses heuristics for input pixel selection.

5: The bias from $1/M$ weights can be removed with *contribution MIS* weights (see Section 7.1.2).

Temporal reuse Effective temporal reuse requires careful tracking of motion vectors, i.e., how pixels move between frames. For temporal reuse, we do RIS between the current pixel and the motion-matched pixel in the previous frame. Correctness again requires proper MIS weights, with the potential challenge that the advanced MIS weights require evaluating last frame's $\hat{p}$, which may require the ability to perform visibility queries in the previous frame's scene (and hence storing the previous frame's ray acceleration structure).

Tip 4.4 We recommend implementing and validating integration with only candidate samples before implementing spatial reuse, spatial reuse before temporal reuse, and temporal reuse first without motion. By validation we mean that averaging a large number of still frames converges to the path tracing ground truth.

4.4 History length

The above method describes a basic version of ReSTIR for direct illumination. However, it has certain, critical, inefficiencies. For example, the temporal reuse assigns equal weights to the previous frame's sample and the new sample. This is not ideal, since it loses roughly 50% of the accumulated history each frame. This can be fixed by weighted MIS, introducing so-called *confidence weights*.

Confidence weights We give samples confidence weights, denoted here c_j, and stored in the pixels' reservoirs. We use the c_j for weighting the samples in resampling,

$$m_i(x) = \frac{c_i \hat{p}_i(x)}{\sum_{j=1}^{M} c_j \hat{p}_j(x)}. \tag{4.7}$$

The more we trust a random variable, the higher its confidence c_i should be. If one of the inputs corresponds to 7 independent samples of a kind, while another corresponds to 2 similar samples, the confidences should be 7 and 2, making MIS favor the more trustworthy sample. The notion of *corresponding to N samples* is often known as *effective sample count*. But, in RIS, we mix samples from different distributions, and effective sample counts are hard to track. We approximate them by simply tracking the number of total input samples the sample has aggregated over its history, summing the confidence weights c_j of all the inputs as the confidence of the result.

When aggregating samples of confidences c_1 and c_2 with RIS, we set the confidence to $c_1 + c_2$. This is, in reality, an upper bound of the effective sample count, but a more accurate estimate is hard to get, hence we use the sum as the confidence. Over multiple frames, these confidences would grow exponentially, each spatial reuse multiplying the confidences. However, only a limited number of new samples are truly added to the pool each frame. Summing confidences is a drastic overestimate and means giving new samples exponentially decreasing relative weights, leading to convergence to a wrong result if not tackled. In practice, the sample confidence is *capped*[6] to a constant that defines the balance between noise and correlation in the final image[7]. Capping the confidence weights is vital to combat the correlations in ReSTIR [4].

6: We commonly cap the sample confidence to somewhere between 5–30. Starting with a cap of 20 is usually good.

7: For historical reasons related to WRS, the confidence weight in the reservoirs is often stored with variable name M, and confidence capping is called *M-capping*. We adopt the convention of denoting confidence by c and its cap by c_{cap}.

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

Algorithm 3: Resampling with Confidence Weights.

```
class Reservoir
    Y, W_Y ← ∅, 0                                   // The output sample
    w_sum ← 0                                       // The sum of weights
    c ← 0                                           // Confidence weight of output
    function update(X_i, w_i, c_i)
        w_sum ← w_sum + w_i
        c ← c + c_i                                 // Update the confidence.
        if rand() < (w_i / w_sum) then
            Y ← X_i
function Resample(M)
    Reservoir r
    for i ← 1 to M do
        generate X_i
        w_i ← m_i(X_i) p̂(X_i) W_{X_i}
        r.update(X_i, w_i, c_i)
    if r.Y ≠ ∅ then
        r.W_Y ← 1 / p̂(r.Y) · r.w_sum
    r.c ← min(r.c, c_cap)
    return r
```

A new independent sample is given confidence 1, and RIS-selecting one from M new samples yields confidence M. Pixels entering the screen as the camera moves (no temporal precedessor) get their M reset to 0, and it often makes sense to reset M also when detecting occlusions or disocclusions. Resetting confidence is allowed only if it can be done based on examining (changes to) the G-buffer; resets depending on sample details leads to bias.

4.5 Advanced topics

Earlier, we proposed using a standard light sampler for direct illumination ReSTIR, with emphasis on easier implementation. A standard light sampler might not be the ideal way of generating samples with ReSTIR, and glossy materials would benefit from BSDF sampling. There is also a conceptual problem in temporal reuse that we will use as a segue to the next chapter.

Improved light sampling Candidate samples can be cheaply generated with, e.g., power-based importance sampling [1]. A light source can be stochastically selected based on its power (total emitted flux over the surface) and a sample point can be picked uniformly from its surface area. Bitterli et al. [1] generate 32 candidate light samples and pick one using WRS, observing real-time performance. The performance can be further improved by precomputing the light samples into "light tiles" shared by screen pixel blocks [5] (to be introduced in Chapter 7).

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

Tip 4.5 ReSTIR consists of multiple parts that all must work. Resist the temptation to optimize, and start by building a principled, simple base system. Once ready and fully validated, improve it piece by piece, thoroughly validating each step.

[5]: Wyman et al. (2021), 'Rearchitecting Spatiotemporal Resampling for Production'

Although each candidate sample is suboptimal, having many of them quickly reduces the variance of the selected sample. Note that power-based light sampling does not consider the properties of the shading point, and the sampling quality can be poor for glossy surfaces. For glossy surfaces, using BSDF sampling as a different strategy turns out to greatly improve the sampling quality [7].

[7]: Bitterli (2022), 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'

Mixing BSDF and light sampling Section 3.3 explains how to do MIS between BSDF and light sampling for the initial candidates, taking M_1 samples with a BSDF importance sampler and M_2 samples with a light sampler. This may also be used with ReSTIR—remember to transform the PDFs into area measure before applying RIS. Proper treatment of glossy materials, however, requires *shift mappings* from the next chapter.

Tip 4.6 We recommend first finalizing and validating the base implementation without BSDF sampling. Debugging multiple parts simultaneously can be hard.

RIS and domains ReSTIR, as we presented it, reuses samples within the same domain. What if objects move, and the domain changes between frames? This is not handled by RIS without extensions! Samples need to be modified to enable reuse between frames[8]. This also requires shift mappings, and an extension of RIS to reuse between domains.

8: Imagine moving a path to the next frame by gluing its vertices to moving objects, matching the triangle index and UV coordinates in both frames. This is an example of a *shift mapping* between frames.

5 Reusing Between Domains

More advanced sample reuse often requires that samples are modified at reuse. The scene changes between frames, and different pixels see different path spaces. Simply reusing vertices without modification does not allow reuse through mirrors or glass. The law of ideal reflection must be obeyed, and paths need to be modified to allow effective reuse.

We now generalize RIS to reuse between domains with *shift mappings*.

5.1 Preliminaries 22
5.1.1 Shift mappings 22
5.1.2 Jacobian determinants . . . 23
5.2 Reusing samples between domains 23
5.3 MIS between domains . . . 24

5.1 Preliminaries

In light transport, shift mappings allow reusing paths between domains, such as path spaces seen by different pixels.

5.1.1 Shift mappings

The term *shift mapping* originates from gradient-domain rendering [38], where samples are moved from one pixel to another—*shifted* on the image plane—for evaluating discrete image gradients $(\Delta I/\Delta x, \Delta I/\Delta y)$; the image is then reconstructed from the combination of color and (discrete) gradient estimates. The path consists of multiple vertices, some of which need to be modified for the path to remain interesting for the other pixel.

[38]: Lehtinen et al. (2013), 'Gradient-Domain Metropolis Light Transport'

Tip 5.1 Implement the reconnection shift first. Always first validate that shifting to the same pixel retains the path and its radiance, and that the Jacobian determinant is then 1:

$$T_{i\to i}(x) = x,$$
$$f(T_{i\to i}(x)) = f(x),$$
$$|T'_{i\to i}(x)| = 1.$$

Report also small discrepancies: path tracers often have bugs only discovered when implementing shift mappings.

Such shift mappings allow reusing paths for other pixels with minimal modifications, taking into account the constraints set by materials such as shiny metals or glass.

A shift mapping T from A to B (e.g., path spaces of different pixels) maps paths in A to paths in B by a relation $y = T(x)$. An example shift mapping is the *reconnection shift* [38] that maps a path to another pixel, reconnecting the deterministic beginning to the same secondary vertex $\mathbf{x}_2$, retaining all free vertices:

$$T_{i\to j}([\mathbf{x}_{i,0}, \mathbf{x}_{i,1}, \mathbf{x}_2, \mathbf{x}_3 \ldots]) = [\mathbf{x}_{j,0}, \mathbf{x}_{j,1}, \mathbf{x}_2, \mathbf{x}_3, \ldots], \tag{5.1}$$

using the notation of Section 4.3. This shift mapping works well for diffuse and rough surfaces, but not for glossy or specular surfaces, as it does not respect the law of ideal reflection, unlike, for example the *half-vector shift* [39], the *random replay shift*, and their hybrids [4, 40].

[39]: Kettunen et al. (2015), 'Gradient-Domain Path Tracing'
[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'
[40]: Hua et al. (2019), 'A Survey on Gradient-Domain Rendering'

Lin et al. [4] formally define shift mappings. The definition encodes the following, partially overlapping properties for a shift mapping T between two domains:

- The shift mapping is deterministic.
- A path may shift to at most one path in the target domain.
- Two paths may not shift to the same path; an inverse shift must exist.

Definition 5.1.1 (shift mapping) *A* shift mapping T *from A to B is a bijective function from a subset $\mathcal{D}(T) \subset A$ to its image $\mathcal{I}(T) \subset B$.*

Note that a shift mapping from A to B is not *always a function from A to B, but from a subset of A, its* domain $\mathcal{D}(T)$*, to a subset of B, its* image $\mathcal{I}(T)$.

- The inverse shift must shift back to the original.
- Not all paths need to be shiftable.

In practical implementations, invertibility is often guaranteed by symmetric shift mappings, making sure that if $y = T_{i\to j}(x)$, then $x = T_{j\to i}(y)$. Invertibility is also required for undefined shifts: if x cannot be shifted with $T_{i\to j}$, then no path y in the target domain is allowed to shift to x.

In practical implementations, the code implementing the shift $T_{i\to j}(x)$ may, during the shifting process, find a show-stopper condition, such as an occlusion in the reconnection shift, forcing $T_{i\to j}$ to halt and return *undefined*.

5.1.2 Jacobian determinants

Imagine mapping close-by real numbers x_1 and x_2 by the same function. The distance between x_1 and x_2 usually grows or shrinks: mapping changes the density of numbers. The local scaling factor is given by the derivative[1].

As shift mappings map paths between domains as $y = T(x)$, they also modify the paths' densities; Jacobian determinants $|T'(x)|$ capture the local scaling factor, which also changes probability densities when passing random variables[2] through mappings. Unbiased contribution weights also change in shift mappings. The Jacobian determinants (often just "Jacobians") are numbers, usually given by relatively simple formulas, that must be included here and there, for example the resampling weights w_i and MIS weights m_i, to retain correctness and help protect from outliers.

The rendering literature knows many good shift mappings along with formulas for their Jacobian determinants [4, 40], precise implementation details, and sample code. We discuss specific shift mappings useful for path reuse in more detail starting in Section 6.4.

1: In multivariate calculus the derivative is called the *Jacobian matrix*, and the scaling factor is its determinant. Our Jacobian determinants have simple geometric formulas.

2: If $Y = T(X)$, then

$$p_Y(Y) = \frac{p_X(X)}{|T'(X)|}. \tag{5.2}$$

Similarly, comparing W_Y to $1/p_Y(Y)$,

$$W_Y = W_X|T'(X)|. \tag{5.3}$$

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'
[40]: Hua et al. (2019), 'A Survey on Gradient-Domain Rendering'

5.2 Reusing samples between domains

We replay our previous exposition of RIS, this time with shift mappings included:

1. Take inputs $(X_1, \dots, X_M)$, each from its own domain Ω_i.
2. Map the samples into the target domain Ω as $Y_i = T_i(X_i)$.
3. Evaluate resampling MIS weights $m_i(Y_i)$ for all Y_i.
4. Evaluate resampling weights $w_i = m_i(Y_i)\,\hat{p}(Y_i)W_{X_i}\,|T_i'(X_i)|$ for all i.
5. Choose Y randomly from the Y_i proportionally to w_i.
6. Evaluate the unbiased contribution weight $W_Y = \frac{1}{\hat{p}(Y)}\sum_{j=1}^{M} w_j$.

This process gives us a sample Y in the target domain Ω that we can use for integration or chaining RIS. Its PDF is approximately proportional to the target function $\hat{p}$; increasingly so with more input samples.

The domains may or may not be the same, and the samples may or may not be to be statistically independent.

W_{X_i}: unbiased contribution weight of X_i in its own domain.

$|T'(X_i)|$ Jacobian determinant of the shift mapping from Ω_i to Ω.

$\Pr[\text{choose } i] = \frac{w_i}{\sum_{j=1}^{M} w_j}$.

Treat W_Y as $1/p(Y)$ but note it is an unbiased estimate, not a function of Y.

Algorithm 4: RIS Between Domains

1 **function** Resample(M)
2 *Reservoir* r
3 **for** $i \leftarrow 1$ **to** M **do**
4 generate X_i `// E.g., take from reservoirs`
5 $Y_i \leftarrow T_i(X_i)$ `// Shift into` Ω
6 $w_i \leftarrow m_i(Y_i)\,\hat{p}(Y_i) W_{X_i}\,|T_i'(X_i)|$ **if** $Y_i \neq \emptyset$ **else** 0 `// 0 if shift failed`
7 r.update(Y_i, w_i, c_i)
8 **if** $r.Y \neq \emptyset$ **then**
9 $r.W_Y \leftarrow \frac{1}{\hat{p}(r.Y)} r.w_{\text{sum}}$
10 $r.c \leftarrow \min(r.c,\ c_{\text{cap}})$
11 **return** r

While the resampling weight formula (step 3) looks different with the addition of the Jacobian determinant, it is essentially the same formula as before, if we use the unbiased contribution weight transformation rule (Equation 5.3) to substitute

$$W_{X_i}|T_i'(X_i)| = W_{Y_i}, \tag{5.4}$$

recovering

$$w_i = m_i(Y_i)\,\hat{p}(Y_i) W_{Y_i}. \tag{5.5}$$

In order to chain RIS or integrate, the inputs, shifted to Ω, should together cover the support of $\hat{p}$. We often guarantee this by letting one of the reuse candidates be a canonical sample with an importance sampler directly targeting $\hat{p}$ and using an identity shift $T_i(x) = x$ with Jacobian determinant $|T_i'(x)| = 1$.

5.3 MIS between domains

The generalized balance heuristic[3] uses the inputs' target functions $\hat{p}_i$ as proxies for the intractable PDFs. Now, the samples X_i come from domains Ω_i, and their target functions $\hat{p}_i$ cannot be evaluated at the shifted $Y_i \in \Omega$, where the MIS weights $m_i(Y_i)$ are evaluated. In other words, the MIS weights require $\hat{p}_i$ evaluations at the input pixels.

Let us, for a second, pretend that we have access to the source PDFs p_i in the input domains. Ideally, we would use the traditional balance heuristic,

$$m_i(y) = \frac{p_{Y_i}(y)}{\sum_{j=1}^{M} p_{Y_j}(y)}, \tag{5.7}$$

where $y \in \Omega$ is in the current domain, and the PDFs p_{Y_i} are for the *mapped* random variables $Y_i = T_i(X_i)$. How can we possibly achieve this?

The PDF transformation rule[4] says $p_{Y_i}(y) = p_{X_i}(x_i)/|T'(x_i)|$, where $y = T_i(x_i)$. This is a good start. What's x_i? We find x_i by shifting y *back* into Ω_i,

3: Generalized balance heuristic in a single-domain case:

$$m_i(x) = \frac{\hat{p}_i(x)}{\sum_{j=1}^{M} \hat{p}_j(x)}. \tag{5.6}$$

4: PDF transformation rule (Equation 5.2):

$$p_Y(Y) = \frac{p_X(X)}{|T'(X)|} \quad \text{if} \quad Y = T(X).$$

$x_i = T_i^{-1}(y)$. How about the division by $|T_i'(x_i)|$? Easy: we already evaluate $x_i = T_i^{-1}(y)$, so we *multiply* by its Jacobian determinant, $|T_i^{-1\prime}(y)|$. This achieves the division by $|T'(x_i)|$, by the inverse function theorem.

Our equation appears more daunting than it truly is:

$$p_{Y_i}(y) = p_{X_i}\left(T_i^{-1}(y)\right)\left|T_i^{-1\prime}(y)\right|. \tag{5.8}$$

Simply shift y back into Ω_i and multiply by the simple formula given for the shift's Jacobian determinant. With the understanding that $p_{Y_i}(y) = 0$ if y cannot be shifted into Ω_i, this allows using the balance heuristic between domains (Equation 5.7)—when the PDFs are known.

We assume the PDFs are not known, and use $\hat{p}_i$ as a proxy for p_{X_i}. As such, we define "$\hat{p}$ from i" [4]. We simply replace p_{X_i} by $\hat{p}_i$, and encode the same zero-condition as:

$$\hat{p}_{\leftarrow i}(y) = \begin{cases} \hat{p}_i\left(T_i^{-1}(y)\right)\left|T_i^{-1\prime}(y)\right|, & \text{if } y \in T_i(\operatorname{supp} X_i) \\ 0 & \text{otherwise} \end{cases}. \tag{5.9}$$

The condition $y \in T_i(\operatorname{supp} X_i)$ simply means that we return 0 if y cannot be shifted into Ω_i, *or* if $x_i = T_i^{-1}(y)$ has zero PDF. When the candidates come from RIS, we can simplify this test into "if cannot shift, return 0", since we recursive guarantee $\hat{p}_i = 0$ exactly when $p_{X_i} = 0$, by giving a canonical sample to RIS if $\operatorname{supp}(\hat{p})$ is not otherwise be covered.

The generalized balance heuristic between multiple domains is then

$$m_i(y) = \frac{\hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} \hat{p}_{\leftarrow j}(y)}, \tag{5.10}$$

and we can also include the confidence weights c_j used by ReSTIR in Section 4.4.

$$m_i(y) = \frac{c_i\, \hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} c_j\, \hat{p}_{\leftarrow j}(y)}. \tag{5.11}$$

Note how Equation 5.10 and 5.11 satisfy the MIS weight requirement

$$\sum_{\substack{i=1 \\ y \in T_i(\operatorname{supp} X_i)}}^{M} m_i(y) = 1. \tag{5.12}$$

with the definition in Equation 5.9. Figure 5.1 shows an illustration.

The generalized balance heuristic is good for a small numbers of candidates, but becomes slow for large numbers due to the total of $O(M^2)$ $\hat{p}_{\leftarrow j}$ terms. We explore more lightweight alternatives in Section 7.1.1 and Section 7.1.2, but we recommend starting using this simple balance heuristic. Correctness first, performance after.

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

Definition 5.3.1 (Canonical sample) *An input $X_i \in \Omega_i$ to RIS is called* canonical *if $\Omega_i = \Omega$, it uses the identity shift map $T_i(x) = x$, uses $\hat{p}_i = \hat{p}$, and covers $\hat{p}$ (i.e., $\operatorname{supp} \hat{p} \subset \operatorname{supp} X_i$).*

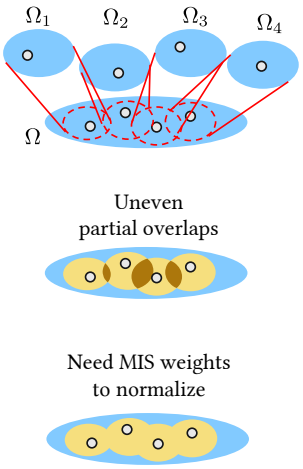

Figure 5.1: Candidate samples from multiple domains can contribute to the same integration at a target domain via shift mapping. But the contribution has to be normalized by MIS weights. Each point in the target domain must be covered exactly once in total.

Tip 5.2 Slow down to speed up! We recommend first implementing a correct, principled and slow version of ReSTIR. Not because bias is always bad, but because this saves a lot of time.

Algorithm 5: Generalized Balance Heuristic

Input : $y \in \Omega$ where $m_i(y)$ is evaluated.
Input : Original $x \in \Omega_i$ that yielded y. We assume $y = T_i(x)$.
Input : Jacobian determinant $\left|T_i'(x)\right|$ of the shift $y = T_i(x)$.
Output: Generalized balance heuristic weight $m_i(y)$.

function pHatFrom(j, y) // $\hat{p}_{\leftarrow j}(y)$ for generic $y \in \Omega$.
  $x_j, Jx_j \leftarrow T_j^{-1}(y), |T_j^{-1\prime}(y)|$ // Shift y into Ω_j and eval Jacobian.
  if $x_j \neq \emptyset$ **then** // If shift succeeded
    return $\hat{p}_j(x_j) \cdot Jx_j$ // $\hat{p}_j(T_j^{-1}(y))\ |T_j^{-1\prime}(y)|$
  return 0

function pHatFrom_opt$(j, x, |T_j'(x)|)$ // $\hat{p}_{\leftarrow j}(y)$ optimized for $y = T_j(x)$
  return $\hat{p}_j(x)/|T_j'(x)|$ // $\hat{p}_j(T_j^{-1}(y))\ |T_j^{-1\prime}(y)|$.

function GenBalanceHeuristic$(i, y; x, \left|T_i'(x)\right|)$ // $m_i(y)$, optimized ($y = T_i(x)$)
  $m_{\text{num}} \leftarrow c_i \cdot$ pHatFrom_opt$(i, x, \left|T_i'(x)\right|)$ // Numerator: $c_i\, \hat{p}_{\leftarrow i}(y)$
  $m_{\text{den}} \leftarrow m_{\text{num}}$ // Case $j = i$. Denominator: $\sum_{j=1}^{M} c_j\, \hat{p}_{\leftarrow j}(y)$
  for $j \leftarrow 1$ **to** M; $j \neq i$ **do** // Cases $j \neq i$
    $m_{\text{den}} \leftarrow m_{\text{den}} + c_j \cdot$ pHatFrom(j, y)
  return $m_{\text{num}}/m_{\text{den}}$

Note:
Return 0 if y is a null sample; this avoids 0/0. See Tip 3.13.

6 ReSTIR Path Tracing

In this chapter, we show how to apply generalized RIS to the path sampling problem for general global illumination, based on the implementation of ReSTIR Path Tracing (ReSTIR PT) [4]. We will first formalize the path sampling problem, followed by applying RIS to a path tree. We then explore shift mapping design and introduce the efficient shift mapping for real-time rendering developed in ReSTIR PT. Finally, we briefly introduce Volumetric ReSTIR [3] which adds support for participating media.

6.1 The path integral 27
6.2 RIS with a path tracer 27
6.3 Reuse path samples 28
6.4 What is a good shift mapping? 28
6.5 Common shift mappings . . 29
6.6 An efficient shift mapping for real-time rendering . . . 30
6.7 Volume rendering 34

6.1 The path integral

A light path can have an arbitrary number of bounces. To extend our path integral to account for global illumination, we need to sample in the union of product spaces $\cup_{D=2}^{\infty} \mathcal{A}^{D-1}$, where $\mathcal{A}$ is the set of all scene surfaces. The full path space integral extends Equation 4.2:

$$L(\mathbf{x}_1 \rightarrow \mathbf{x}_0) = \sum_{D=2}^{\infty} \int_{\mathcal{A}^{D-1}} \Big(\prod_{j=1}^{D-1} f_s(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}) G(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) V(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) \Big) L_e(\mathbf{x}_D \rightarrow \mathbf{x}_{D-1}) \, \mathrm{d}\mathbf{x}_2 \dots \mathrm{d}\mathbf{x}_D \, . \tag{6.1}$$

A path sample of $D-1$ bounces can be written in the following tuple form in area measure:

$$[\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_D] \tag{6.2}$$

6.2 RIS with a path tracer

With next-event estimation (NEE) at every path vertex, a path tracer usually creates a path tree. We want to use RIS to pick one path from the entire path tree, which we can then input into the ReSTIR pipeline.

Assume all light vertices are sampled by NEE, and label paths (from the same path tree) from 1 to k bounces as $x_1, x_2, \dots x_k$, we can still apply the RIS resampling weight formula: $w_i = m_i(x_i)\hat{p}(x_i)W_{x_i}$. Note that $m_i(x_i) = 1$ here, since each path sample is responsible for a different path subspace and different path subspaces are disjoint (a sampling technique for one subspace will have zero PDF for any sample outside the subspace). It is common to set $\hat{p}(x_i) = f(x_i)$, the path contribution [2, 4]. And it is easy to know that $W_{x_i} = 1/p(x_i)$, the reciprocal of path PDF. Both $f(x_i)$ and $p(x_i)$ are easily obtained from a path tracer.

[2]: Ouyang et al. (2021), 'ReSTIR GI: Path Resampling for Real-Time Path Tracing'
[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

It is common for a path tracer to MIS NEE light sample with light samples found by next hits of scattered rays (BSDF sampling). To account for this in RIS, both NEE and BSDF light samples are used as candidate paths, doubling the candidate samples. Each $m_i(x_i)$ now accounts for the other light sampling technique of the same length and is generally less than 1[1].

1: Alternatively, the path integral can be partitioned into two integrals for the two light sampling techniques with the light sampling MIS weights being part of path contribution $f(x_i)$ for each technique, which preserves $m_i(x_i) = 1$ in RIS.

6.3 Reuse path samples

The simplest way to reuse samples is to use identity shift (in area measure):

$$T([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D]) = [\mathbf{y}_0, \mathbf{y}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D] \,. \tag{6.3}$$

This resembles the direct lighting case (Section 4.3), but the whole sequence of vertices $[\mathbf{x}_2, \ldots, \mathbf{x}_D]$ is reused.This shift mapping is used by ReSTIR GI [2]. A proper implementation requires connecting from $\mathbf{y}_1$ to $\mathbf{x}_2$, which involves the re-evaluation of two BSDFs, a geometry factor, and shooting a shadow ray. A biased implementation may prioritize computational speed or minimal memory usage at the expense of render accuracy.

[2]: Ouyang et al. (2021), 'ReSTIR GI: Path Resampling for Real-Time Path Tracing'

For example, ReSTIR GI precomputes the outgoing radiance along $\overline{\mathbf{x}_2\mathbf{x}_1}$ when the sample is produced by a path tracer, and assume it is unchanged along the reconnected direction $\overline{\mathbf{x}_2\mathbf{y}_1}$, during reuse. Obviously, this is only true for a very limited set of materials like Lambertian diffuse material.

Biased implementation like ReSTIR GI cannot generate faithful results if $\mathbf{x}_2$ is specular. With an unbiased implementation, the average of independent renders will converge to the ground truth. But reusing paths with specular vertices can increase the variance instead of reducing it.

6.4 What is a good shift mapping?

What's an ideal shift mapping? Assuming all pixels have precedent samplers (prior to reuse) with comparable quality, when shifting from pixel i to pixel j, we want the shift mapping to create shifted samples $y = T(x)$ such that its distribution is as good as a sample produced by pixel j's sampler. If we assume that each pixel has a low variance importance sampler for the target function , i.e. $p_X \approx \bar{p}$, an ideal shift mapping should make the target PDF of the shifted sample approximately equal to the original target PDF (with density transformation) [41], i.e. [2]

$\bar{p}(x) = \hat{p}(x) / \int_\Omega \hat{p}(x) \mathrm{d}x$

[41]: Tokuyoshi (2023), 'Efficient Spatial Resampling Using the PDF Similarity'

$$\bar{p}_j(T(x)) \left| \frac{\partial T}{\partial x} \right| \approx \bar{p}_i(x) \,. \tag{6.4}$$

2: Lin et al. [4] proposes a stronger (but not necessary) condition: $\hat{p}_j(T(x)) \approx \hat{p}_i(x)$ and $\left|\frac{\partial T}{\partial x}\right| \approx 1$.

With the identity shift in area measure (simply reusing all free path vertices), this is likely to be true when reusing a distant $\mathbf{x}_2$ on a diffuse surface from a neighboring pixel.

But $\bar{p}_j(T(x))$ and $\bar{p}_i(x)$ can be extremely different for x with $\mathbf{x}_1/\mathbf{y}_1$ and/or $\mathbf{x}_2$ on specular surfaces. A path sample with significant contribution on

the source pixel may end up having zero or near-zero path contribution when shifted to another pixel due to the delta or low roughness specular material (see Figure 6.1 for an illustration). Even with diffuse material, the situation that reconnection segment $\overline{\mathbf{x}_2\mathbf{y}_1}$ having a drastically different length than $\mathbf{x}_2\mathbf{x}_1$ also breaks Equation 6.4 by huge or tiny ratios of the geometry terms. A good shift mapping should avoid these two scenarios, which means all vertices should not be simply reused. Still, we want to reuse as many vertices as possible for the reason that shifted path segments that are coincident with the corresponding original segments yields identity Jacobians and equal terms in $\hat{p}$.

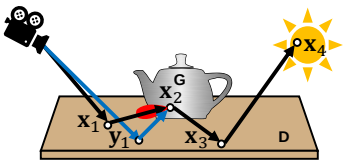

Figure 6.1: An example case where reconnection shift fails. Note that the shifted path (blue) connects to a glossy surface which gives near-zero BSDF value in the new direction.

6.5 Common shift mappings

Shift mappings have been studied extensively in gradient domain rendering [38–40, 42]. Assuming smooth local variation of image intensity, for nearby pixels i and j, $\hat{p}_i$ and $\hat{p}_j$ should have similar normalization factors. With this assumption and $\hat{p} \approx f$, Equation 6.4 can be equivalently tested using

$$f_j(T(x))\left|\frac{\partial T}{\partial x}\right| \approx f_i(x)\ , \tag{6.5}$$

which is the same condition used by gradient domain rendering [39]. This means that shift mappings developed for gradient domain rendering can also be used for path resampling.

[38]: Lehtinen et al. (2013), 'Gradient-Domain Metropolis Light Transport'
[39]: Kettunen et al. (2015), 'Gradient-Domain Path Tracing'
[40]: Hua et al. (2019), 'A Survey on Gradient-Domain Rendering'
[42]: Manzi et al. (2014), 'Improved Sampling for Gradient-Domain Metropolis Light Transport'

For an example, we examine the shift mapping introduced in gradient domain path tracing [39]. To shift a path, gradient domain path tracing sequentially constructs the offset path vertex by vertex by copying the tangent-space half-vector at the corresponding base path vertex to trace next offset path vertex using the conforming reflection or refraction direction. This is repeatedly performed until two consecutive diffuse vertices (classified using a threshold on material roughness) $\mathbf{x}_{k-1}, \mathbf{x}_k$ on the base path are encountered, where the offset path vertex $\mathbf{y}_{k-1}$ connects to the base path vertex $\mathbf{x}_k$. Due to the invertibility requirement of shift mappings, $\mathbf{y}_{k-1}$ also needs to be diffuse for the shift to be successful.

Half-vector shift tends to preserve the path throughput through (near-)specular vertices, as the importance-sampled specular reflection directions are centered around the perfect mirror reflection direction. An in-depth analysis is provided by Kaplanyan et al. [43] to show that the half-vector parameterization yields a mostly smooth path throughput function for glossy materials. However, as the bounce count increases, the shifted vertices tend to diverge from the original path, enlarging the difference of path throughputs. Therefore, it is still desirable to connect back to the base path when the material types are proper.

[43]: Kaplanyan et al. (2014), 'The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation'

Noticeably, the shift mapping of Kettunen et al. [39] is comprised of two local shift decisions: half-vector copy and vertex reconnection local shift mapping. Lin et al. [4] provide an overview of common local shift decisions. In comparison, global methods like manifold exploration [44] allows more

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'
[44]: Jakob et al. (2012), 'Manifold Exploration'

tightly preserving the path contribution. But global operations like solving the specular chain between two diffuse vertices are more computationally intensive[3].

3: Since manifold exploration can solve the specular chain between two diffuse vertices, the two diffuse vertices in the shifted path need not to be consecutive, making it share more vertices with the base path.

6.6 An efficient shift mapping for real-time rendering

To better combine with ReSTIR's real-time rendering goal, ReSTIR PT [4] presents an efficient shift mapping suitable for the GPU. The shift mapping postpones the reconnection similar to Gradient Domain Path Tracing [39], but features three differences.

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

[39]: Kettunen et al. (2015), 'Gradient-Domain Path Tracing'

- It precomputes the reconnection vertex on the base path and uses random replay [40] to shift earlier path segments so that the base path does not have to be stored. Random replay copies the base path's random numbers at each bounce to re-trace the next bounce with the method used by the base path. It usually makes decisions similar to copying the half-vector or direction (depending on the BSDF type), or a light source's position in the case of next-event-estimation.
- It uses an additional distance condition similar to Manzi et al. [42] to avoid creating short reconnection segments.
- It classifies a vertex using only roughness of the sampled lobe, optimizing resampling on multi-layer materials.

[40]: Hua et al. (2019), 'A Survey on Gradient-Domain Rendering'

[42]: Manzi et al. (2014), 'Improved Sampling for Gradient-Domain Metropolis Light Transport'

This is called *hybrid shift* in ReSTIR PT. Note that this shift mapping can be integrated nicely with a modern path tracer and only requires constant storage per pixel – only a reconnection vertex and a random-number-generating seed in addition to other reservoir data like the unbiased contribution weight. Compared to reconnection shift discussed in Section 6.4, hybrid shift can significantly improve the quality of glossy and refractive material.

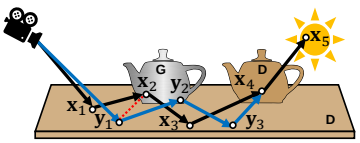

Figure 6.2: A hybrid shift mapping. The base path selects $\mathbf{x}_4$ for reconnection, since both $\mathbf{x}_3$ and $\mathbf{x}_4$ are rough ($k = 4$). The offset path copies the random numbers of the base at $\mathbf{x}_1$ and $\mathbf{x}_2$ to construct similar scatter directions for $\mathbf{y}_1$ 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 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. Figure taken from 'Generalized Resampled Importance Sampling' [4].

To ensure a successful implementation of hybrid shift in ReSTIR PT, it is crucial to focus on the following key details:

Ensuring Invertibility For a $\mathbf{x}_k$ on the base path to be connectible, it needs to satisfy two conditions:

- Distance Condition:

$$\min(||\mathbf{x}_k - \mathbf{x}_{k-1}||, ||\mathbf{x}_k - \mathbf{y}_{k-1}||) \geq d_{\min} \tag{6.6}$$

- Roughness Condition:

$$\min(\alpha_{\mathbf{x}_{k-1}}(\ell_{k-1}), \alpha_{\mathbf{y}_{k-1}}(\ell'_{k-1}), \alpha_{\mathbf{x}_k}(\ell_k)) \geq \alpha_{\min} \tag{6.7}$$

  Note that $\ell_{k-1}, \ell_k, \ell'_{k-1}$ are sampled lobes (e.g. Lambertian diffuse, Microfacet glossy with GGX distribution) on $\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{y}_{k-1}$, respectively. $\alpha_{\mathbf{x}}(\ell)$ measures the roughness of the lobe ℓ (usually in a $[0, 1]$ range for specular material and it can be set to a large value for diffuse

material) at vertex $\mathbf{x}$. Some vertices in the sampled path may contain more than one lobe[4] (like vertices that samples lights for NEE and evaluates all lobes). In that case, ReSTIR PT picks the lobe ℓ that maximizes $\alpha(\ell)$.

4: We will introduce the concept of extended path sample used in ReSTIR PT shortly.

When the base path is initially traced by a path tracer, the vertex $\mathbf{x}_k$ with the smallest k ($k \geq 2$) that satisfies

$$||\mathbf{x}_k - \mathbf{x}_{k-1}|| \geq d_{\text{min}} \tag{6.8}$$

and

$$\min(\alpha_{\mathbf{x}_{k-1}}(\ell_{k-1}), \alpha_{\mathbf{x}_k}(\ell_k)) \geq \alpha_{\text{min}} \tag{6.9}$$

is stored as the reconnection vertex.

When shifted, the offset path is generated using random replay until $\mathbf{y}_{k-1}$ which is then connected to $\mathbf{x}_k$. To be invertible, it is also required that

$$||\mathbf{x}_k - \mathbf{y}_{k-1}|| \geq d_{\text{min}} \tag{6.10}$$

and[5]

$$\min(\alpha_{\mathbf{y}_{k-1}}(\ell_{k-1}), \alpha_{\mathbf{x}_k}(\ell_k)) \geq \alpha_{\text{min}} \ . \tag{6.11}$$

5: ℓ_{k-1} is copied to the offset path such that $\ell'_{k-1} = \ell_{k-1}$

Importantly, there cannot be a $k' < k$ that satisfies the same conditions, i.e. $||\mathbf{y}_{k'} - \mathbf{y}_{k'-1}|| \geq d_{\text{min}}$ and $\min(\alpha(\ell'_{k'-1}), \alpha(\ell'_{k'})) \geq d_{\text{min}}$ cannot be both true for a $k' < k$. Otherwise, the offset path could have computed a different k (had it been the base path) and the invertibility is broken. Non-invertible samples get zero weight in RIS.

Path Samples with Lobe/Technique Tags To have a unique mapping between path space and primary sample spaces (for the random number sequence), ReSTIR PT extends path samples with lobe and light sampling technique tags. An extend path sample is represented as

$$\bar{x} = [\mathbf{x}_0, (\mathbf{x}_1, \ell_1), (\mathbf{x}_2, \ell_2), \ldots, (\mathbf{x}_{D-1}, \ell_{D-1}), \mathbf{x}_D] \tag{6.12}$$

where $0 \leq \ell_j \leq N_{\text{lobe}}$ is an index representing the sampled lobe[6] at vertex $\mathbf{x}_j$. Specially, when the light is sampled by NEE, the lobe tag $\ell_{D-1} = N_{\text{lobe}}$ is set to indicate the case, and all lobes are contained in $\mathbf{x}_{D-1}$. With these lobe and technique tags, random replay can produce the exact same (sub)path produced by a path tracer [7].

6: N_{lobe} is the total number of lobes in the scene.

7: The actual implementation may need to skip random numbers used towards other parts of the path tree than the path being reused.

When reconnecting to $\mathbf{x}_k$, it is important to copy the lobe index ℓ_{k-1} to ensure bijection. If such lobe does not exist on $\mathbf{y}_{k-1}$, then the shift fails.

Additional math notes: Denote the set of all possible lobe/technique index sequences $\bar{\ell} = [\ell_1, \ell_2, \ldots]$ for length-$(D+1)$ paths as $\mathcal{L}_D$, and the set of lobe-tagged length-$(D+1)$ paths as $\bar{\Omega}_D$, the path integral can be rewritten into the following form:

$$I = \sum_{D=1}^{\infty} \int_{\bar{\Omega}_D} \bar{f}(\bar{x}) \, \mathrm{d}\bar{x} = \sum_{D=1}^{\infty} \sum_{\bar{\ell} \in \mathcal{L}_D} \int_{\mathcal{A}^D} m_t(x) f_{\bar{\ell}}(x) \, \mathrm{d}x, \tag{6.13}$$

where f_ℓ is a partial path contribution only evaluating the sampled BSDF at each path vertex, $m_t(x)$ is the light sampling technique MIS weight ($t \in \{0, 1\}$ indicates whether the light vertex is sampled by BSDF or NEE, and is inferred from ℓ_{D-1}). From this relationship, we can see that the integrand for an extended path sample $\bar{f}(\bar{x}) = m_t(x) f_\ell(x)$ contains the light sampling technique MIS weight and only uses partial path contribution.

Primary Sample Spaces ReSTIR PT uses primary sample space parameterization for the paths. There are two benefits:

- Path integrands can be expressed as sampled path throughput (product of "f/p" terms at each bounce), which can be directly provided by a path tracer. And initial candidate path PDF is always 1. This also prevents potential floating point number overflows by tracking f and p separately.
- PSS is the more convenient choice for the shift mapping. The random replay part of the shift mapping has an identity Jacobian determinant. Only the reconnection phase computes terms for the Jacobian of the full shift. (If not using PSS, each random replayed path vertex needs to compute terms for the Jacobian.)

In PSS, the path integral can be expressed as

$$I = \sum_{D=1}^{\infty} \int_{\mathcal{U}_D} F(\bar{\mathbf{u}}) \, \mathrm{d}\bar{\mathbf{u}} \tag{6.14}$$

where $\bar{\mathbf{u}}$ is a random number sequence suitable for producing length-$(D+1)$ paths, $\mathcal{U}_D = [0, 1]^{\mathcal{N}(D)}$ is a unit hypercube with $\mathcal{N}(D)$ (length of $\bar{\mathbf{u}}$) dimensions, and $F(\bar{\mathbf{u}}) = \bar{f}(\mathcal{X}(\bar{\mathbf{u}}))/p(\mathcal{X}(\bar{\mathbf{u}}))$ is the integrand ($\mathcal{X}$ maps $\bar{\mathbf{u}}$ to the corresponding extended path sample $\bar{x}$, and $p(\bar{x})$ is the PDF of the path sample in path space parameterization). Note the one-to-one correspondence between Equation 6.14 and the left-hand side of Equation 6.13.

In following text, we provide a deriviation of the hybrid shift Jacobian determinant PSS in parameterization.

For shifting base path x to offset path y, we denote by ω^x_{k-1} the unit vector from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$, and the corresponding random numbers leading from vertex $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ by $\bar{\mathbf{u}}^x_{k-1}$.

When using the common solid angle parametrization, the Jacobian for the reconnection shift is (e.g., [39])

[39]: Kettunen et al. (2015), 'Gradient-Domain Path Tracing'

$$\left| \frac{\partial \omega^y_{k-1}}{\partial \omega^x_{k-1}} \right| = \left| \frac{\cos \theta^y_k}{\cos \theta^x_k} \right| \frac{||\mathbf{x}_k - \mathbf{x}_{k-1}||^2}{||\mathbf{x}_k - \mathbf{y}_{k-1}||^2}, \tag{6.15}$$

for $\theta_k^\bullet$ the angle between $\omega_{k-1}^\bullet$ and the geometric surface normal at $\mathbf{x}_k = \mathbf{y}_k$.

In PSS, the Jacobian determinant of hybrid shift is equal to the local Jacobian determinant of the reconnection step. The shift mapping changes the random numbers for both $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$. We have

$$\left|\frac{\partial \bar{\mathbf{u}}^y}{\partial \bar{\mathbf{u}}^x}\right| = \left|\frac{\partial \bar{\mathbf{u}}_{k-1}^y}{\partial \bar{\mathbf{u}}_{k-1}^x}\right| \left|\frac{\partial \bar{\mathbf{u}}_k^y}{\partial \bar{\mathbf{u}}_k^x}\right| \tag{6.16}$$

(the $\left|\frac{\partial \bar{\mathbf{u}}_k^y}{\partial \bar{\mathbf{u}}_k^x}\right|$ term is dropped when $\mathbf{x}_k = \mathbf{y}_k$ is a light vertex ($\mathbf{x}_{k+1}$ does not exist).) where

$$\left|\frac{\partial \bar{\mathbf{u}}_{k-1}^y}{\partial \bar{\mathbf{u}}_{k-1}^x}\right| = \left|\frac{\partial \bar{\mathbf{u}}_{k-1}^y}{\partial \omega_{k-1}^y}\right| \left|\frac{\partial \omega_{k-1}^y}{\partial \omega_{k-1}^x}\right| \left|\frac{\partial \omega_{k-1}^x}{\partial \bar{\mathbf{u}}_{k-1}^x}\right| = \frac{p_{(\omega,\ell)_{k-1}^y}(\mathbf{y}_k)}{p_{(\omega,\ell)_{k-1}^x}(\mathbf{x}_k)} \left|\frac{\partial \omega_{k-1}^y}{\partial \omega_{k-1}^x}\right| \tag{6.17}$$

and

$$\left|\frac{\partial \bar{\mathbf{u}}_k^y}{\partial \bar{\mathbf{u}}_k^x}\right| = \left|\frac{\partial \bar{\mathbf{u}}_k^y}{\partial \omega_k^y}\right| \left|\frac{\partial \omega_k^x}{\partial \bar{\mathbf{u}}_k^x}\right| = \frac{p_{(\omega,\ell)_k^y}(\mathbf{y}_{k+1})}{p_{(\omega,\ell)_k^x}(\mathbf{x}_{k+1})}\,, \tag{6.18}$$

where

$p_{(\omega,\ell)_{k-1}^x}(\mathbf{x}_k) = p_{\omega_{k-1}^x}(\mathbf{x}_k) \cdot p_{\ell_{k-1}^x}$ is the joint PDF in solid angle measure for sampling lobe ℓ_{k-1}^x [8] and direction ω_{k-1}^x on path x (the PDF and PMF may implicitly depend on other path vertices, for example, $\mathbf{x}_{k-2}$, as common BSDF sampling procedures do) and the other terms are similarly defined on other vertices.

Note that $\omega_k^x = \omega_k^y$ and $\ell_{k-1}^x = \ell_{k-1}^y$ by our reconnection definition (assuming the scene is static). Although $\omega_k^x = \omega_k^y$, their sampling PDFs are different due to different outgoing directions towards the previous path vertices.

8: If the path vertex $\mathbf{x}_k$ is a light vertex directly sampled by NEE, then $p_{w_{k-1}^y}(\mathbf{y}_k)$ is the light sampling PDF converted to solid angle measure and $p_{\ell_{k-1}^y} = 1$ (similar for $\mathbf{x}_{k+1}$ if $\mathbf{x}_{k+1}$ is a NEE light vertex).

Reservoir and implementation details in ReSTIR PT. Algorithm 6 details the reservoir structure used by ReSTIR PT. In particular, the path sample Y is represented by the reconnection vertex, the random number generator (RNG) seeds, and some other information that are sufficient for the shift mapping and sample evaluation. We explain these class members together with a typical resampling process (Assume the reconnection vertex $\mathbf{y}_k$ is not a light vertex. The other case is simpler and can be processed similarly.) as follows:

The RNG seed ξ_1 generates the offset sub-path $[\mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_{k-1}]$ with a throughput β. With $\mathbf{y}_k = \mathbf{x}_k$ computed using the triangle Id (Δ) and the barycentrics tuple (λ_1, λ_2) [9], and the incident direction ω (same as ω_k^x) with corresponding radiance estimate L, and the lobe indices ℓ_k, ℓ_{k+1}, the reconnected PSS path contribution can be evaluated as $F(Y) = \hat{p}(Y) = \beta \cdot m_t(\omega_k^y) f_{s,\omega_{k-1}^y}(\mathbf{y}_k)/p_{\omega_{k-1}^y}(\mathbf{y}_k) \cdot f_{s,\omega_k^y}(\mathbf{y}_{k+1})/p_{\omega_k^y}(\mathbf{y}_{k+1}) \cdot L$ [10], where $m_t(\boldsymbol{\omega}_k^y)$ is the MIS weight for sampling ω_k^y using technique t ($t \in \{0, 1\}$ indicates BSDF or NEE and is inferred from the lobe index). $m_t(\boldsymbol{\omega}_k^y) = 1$ if $\mathbf{y}_{k+1}$ is not a light vertex, and is the traditional light sampling MIS weight

9: If the scene changes, $(\Delta, \lambda_1, \lambda_2)$ automatically maps $\mathbf{y}_k$ to the potentially animated triangle. But a temporal shift mapping is required to update L due to potential geometry or lighting changes. One way is to reuse the world space direction ω to find vertex $\mathbf{y}_{k+1}$ and reuse RNG seed ξ_2 recorded on $\mathbf{x}_{k+1}$ to generate the rest of the path $[\mathbf{y}_{k+2}, ...]$.

10: If $\mathbf{y}_k$ is the light vertex, the path contribution is written with reduced terms $\beta \cdot m_t(\omega_{k-1}^y) f_{s,\omega_{k-1}^y}(\mathbf{y}_k)/p_{\omega_{k-1}^y}(\mathbf{y}_k) \cdot L$.

Algorithm 6: Reservoir in ReSTIR PT.

```
class Reservoir
    struct Y
        struct RcVertex
            ω, L, (Δ, λ1, λ2), ℓ_{k−1}, ℓ_k
        RcVertex, ξ1, ξ2, k, J
    Y, W_Y ← ∅, 0                          // The output sample
    w_sum ← 0                              // The sum of weights
    c ← 0                                  // Confidence weight of output
```

otherwise. The Jacobian for the shift mapping can be computed using Equation 6.16. Note that the reservoir stores the part associated with the base path ($J = p_{\omega^x_{k-1}}(\mathbf{x}_k)\cdot|\cos\theta^x_k|/||\mathbf{x}_k-\mathbf{x}_{k-1}||^2\cdot p_{\boldsymbol{\omega}^x_k}(\mathbf{x}_{k+1})$) which is updated after each RIS to avoid the duplicated effort of re-computing it in the next resampling.

6.7 Volume rendering

ReSTIR can also be extended to handle general light transport with participating media [3]. With participating media, the per-bouce integration domain becomes the union of surface area and volume, i.e. $\mathcal{M} = \mathcal{A} \cup \mathcal{V}$.

[3]: Lin et al. (2021), 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'

The pixel intensity can be written as an integral of measurement contribution

$$I_j = \sum_{D=1}^{\infty} \int_{\mathcal{M}^{D+1}} W_e^{(j)}(\mathbf{x}_1 \to \mathbf{x}_0) T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \bar{G}(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \Big(\prod_{j=1}^{D-1} \bar{f}_s(\mathbf{x}_{j+1} \to \mathbf{x}_j \to \mathbf{x}_{j-1}) \bar{G}(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) T(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) \Big) \bar{L}_e(\mathbf{x}_D \to \mathbf{x}_{D-1}) \, \mathrm{d}\mathbf{x}_0 \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2 \ldots \mathrm{d}\mathbf{x}_D \,, \tag{6.19}$$

where $W_e^{(j)}$ is the pixel response function of pixel j (importance function multiplied with pixel filter) and $\mathbf{x}_0$ is a point on the camera sensor (image plane).

Assume a pinhole camera and we are interested only in the radiance arriving at a specific subpixel location, we can write an equation similar to Equation 4.2

$$L(\boldsymbol{\omega}_0 \to \mathbf{x}_0) = T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1^s)(L_e(\mathbf{x}_1^s \to \mathbf{x}_0) + P(\mathbf{x}_0, \mathbf{x}_1^s)) + \int_0^s T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)(\sigma_a(\mathbf{x}_1) L_e^m(\mathbf{x}_1 \to \mathbf{x}_0) + P(\mathbf{x}_0, \mathbf{x}_1)) \, \mathrm{d}z_1 \,, \tag{6.20}$$

where z_1 is the collision distance with the relationship $\mathbf{x}_1 = \mathbf{x}_0 + z_1\boldsymbol{\omega}_0$ ($\boldsymbol{\omega}_0$ is determined by the subpixel location), $\mathbf{x}_1^s$ is the closest opaque surface intersected with $s = ||\mathbf{x}_1^s - \mathbf{x}_0||$, and

$$P(\mathbf{x}_0, \mathbf{x}_1) = \sum_{D=2}^{\infty} \int_{\mathcal{M}^{D-1}} \Big(\prod_{j=1}^{D-1} \bar{f}_s(\mathbf{x}_{j+1} \to \mathbf{x}_j \to \mathbf{x}_{j-1}) \\ \bar{G}(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) T(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) \Big) \bar{L}_e(\mathbf{x}_D \to \mathbf{x}_{D-1}) \, \mathrm{d}\mathbf{x}_2 \dots \mathrm{d}\mathbf{x}_D \tag{6.21}$$

σ_t := extinction coefficient
σ_s := scattering coefficient
σ_a := absorption coefficient
$\sigma_t(\mathbf{x}) = \sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x}), \forall \mathbf{x}$

is a shorthand for the "in-scattered" path contribution from a point $\mathbf{x}_1$ to $\mathbf{x}_0$. $\bar{f}_s, \bar{G}, \bar{L}_e$ are generalized bidirectional scattering function, geometry term, and vertex emission, respectively: $\bar{f}_s = \sigma_s(\mathbf{x}_j)\rho(\mathbf{x}_{j+1} \to \mathbf{x}_j \to \mathbf{x}_{j-1})$ if $\mathbf{x}_j$ is in volume and $\bar{f}_s = f_s$ if $\mathbf{x}_j$ is on surface, $\bar{G} = 1/||\mathbf{x}_j - \mathbf{x}_{j+1}||^2$ if $\mathbf{x}_j$ is in volume and $\bar{G} = G$ is $\mathbf{x}_j$ is on surface, $\bar{L}_e(\mathbf{x}_D \to \mathbf{x}_{D-1}) = \sigma_a(\mathbf{x}_D) L_e^m(\mathbf{x}_D \to \mathbf{x}_{D-1})$ if $\mathbf{x}_D$ is in volume and $\bar{L}_e = L_e$ if $\mathbf{x}_D$ is on surface.

The main difference from surface-only light transport is that a path vertex can live in the entire 3D space instead of only 2D manifolds in 3D. Thus, path sampling usually needs to include the sampling of collision distance. In addition, the integrand includes transmittance terms which may not be obtainable in closed form[11]. Note that

$$T(\mathbf{x} \leftrightarrow \mathbf{y}) = \mathrm{e}^{-\int_0^z y \sigma_t(\mathbf{x} - y\boldsymbol{\omega}) \mathrm{d}y} \tag{6.22}$$

defines the transmittance between $\mathbf{x}$ and $\mathbf{y}$ along direction $\boldsymbol{\omega} = \frac{\mathbf{y}-\mathbf{x}}{z}$ where $z = ||\mathbf{y} - \mathbf{x}||$. Because the transmittance contains an integral which is usually costly to evaluate, Volumetric ReSTIR [3] uses a simplified $\hat{p}$ which has transmittance terms approximated by ray marching. The ray marching step size controls the accuracy of approximation and trades between speed and sampling variance[12]. Besides, Volumetric ReSTIR evaluates transmittance in $\hat{p}$ with low resolution volumes to minimize memory cost. A piece-wise constant, low resolution volume is used for initial path sampling so that the transmittance function can be inverted analytically to produce distance samples with closed-form path PDFs. By reserving the accurate evaluation of path transmittance for final shading, Volumetric ReSTIR achieves efficient resampling, allowing low-noise, interactive volume rendering in complex lighting scenarios.

Because $\mathbf{x}_1$ depends on the sampled collision distance, Volumetric ReSTIR copies the collision distance z_1 to create the shifted $\mathbf{x}_1$ in a different pixel[13]. For the remaining path, Volumetric ReSTIR presented two reuse methods. The first method (vertex reuse) copies the vertex sequence $[\mathbf{x}_2, \dots, \mathbf{x}_D]$ like reconnection shift in ReSTIR PT. The second method (direction reuse) copies the scattering direction and collision distance of all path segments to retrace the shifted path except for the last one where the light vertex is reconnected, i.e. the sequence $[z_1, \boldsymbol{\omega}_1, z_2, \boldsymbol{\omega}_2, \dots, z_{D-1}, \mathbf{x}_D]$.

While vertex reuse has performance advantage, Lin et al. [3] observed excessive amount of fireflies caused by geometric singularity, so they opted for the slower direction reuse by default. To improve direction reuse, ideas from ReSTIR PT's hybrid shift could be combined: for example, reconnection should happen whenever the reconnection segment length is longer than the distance threshold. Since a distance sample can be generated using only one random number (sampling the transmittance and solve for

11: In some cases where the spatial distribution σ_t is piecewise "simple" (e.g. piecewise constant or trilinearly interpolated by neighboring grid points), the close form can be computed.

[3]: Lin et al. (2021), 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'

12: For general volumes, collision distances can be sampled according to transmittance by delta tracking [34] but the PDFs are unknown.

13: If $\mathbf{x}_1$ is on a surface, the shift mapping in Volumetric ReSTIR puts the shifted $\mathbf{x}_1$ on the closest surface in the target pixel. The same applies for the remaining path vertices. Bijection is maintained by requiring both the original and the shifted vertex to be on the closest surfaces or in the (unoccluded) media.

the distance), replacing distance/direction reuse with random replay is also feasible.

Making ReSTIR fast 7

One key appeal of resampling, and ReSTIR in particular, is it offers high quality sampling for real-time rendering, so high-performing implementations are vital for real applications. But before talking about specific optimizations, let's step back and ask *what* we should optimize.

After all, ReSTIR is a general sampling technique. Sampling techniques are usually evaluated based on *sampling efficiency*, combining sample cost and quality into a single metric. Because ReSTIR is based on resampling, the efficiency of the sampler is affected by what neighbors it chooses to reuse and the choice of MIS weights. In addition, efficiency can be improved by low-level optimizations, improving sample quality at a given performance, or both. As a result, we categorize the optimization techniques into *sampler optimization* and *low-level optimization*.

7.1 Sampler optimization . . . 37
7.1.1 Neighbor rejection as approximate "MIS weights" . 38
7.1.2 Contribution MIS weights 38
7.1.3 Pairwise MIS weights . . . 39
7.1.4 Biased MIS Weights 41
7.2 Low-level optimization . . 42
7.2.1 Sample tiling in ReSTIR DI 43
7.2.2 Lighting with many analytic light types 45
7.2.3 Accelerating hybrid shift . 45

7.1 Sampler optimization

One way to look at RIS and ReSTIR: fundamentally, they are simply ways of combining multiple estimators together using MIS weights. Each pixel we borrow from is actually a different estimator we can use to draw samples for our current pixel. Frequently, people learn about MIS by exploring Veach et al.'s [32] sample combination of BSDF and light samples, but MIS can be used to combine almost any estimators together, including our strange resampled-neighbor estimators.

[32]: Veach et al. (1995), 'Optimally Combining Sampling Techniques for Monte Carlo Rendering'

An easily overlooked, but important point is that combining estimators with MIS is not *guaranteed* to improve sampling quality. Generally, combining BSDF and light samples is almost always a win, so it is easy to forget this point. But when reusing neighbor pixel samples, it is fairly easy to select neighbors that are *horrible* estimators for your current pixel.

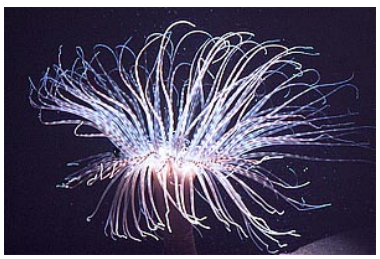

Figure 7.1: Sea anemone, with spindly features where neighbor pixels might be poor estimators for the current pixel. (Image CC-by-SA-3.0, Massimiliano De Martino).

Consider the sea anemone in Figure 7.1, where nearby neighbors may have surface normals in, essentially, opposite directions. The set of paths that contribute to both these neighbors is largely empty. Because of this, reusing samples across these neighbors is unlikely to prove beneficial. In fact, such reuse typically increases noise.

Defining a way to skip reuse from obviously irrelevant neighbors can, thus, provide an efficiency gain.

This need not bias the result, if done carefully. For instance, skipping reuse if the surface normals at primary hits vary too much is fine. Heuristics for skipping reuse are unbiased if they do not look at individual samples or weights. Reasoning about their domains is fine, but making decisions on specific samples generally conditions them, moving reuse into conditional probability spaces.

Heuristics we have found useful include ensuring surface normals, depths, and material properties do not vary significantly between reused pixels.

7.1.1 Neighbor rejection as approximate "MIS weights"

As discussed repeatedly, e.g., in Section 2.2 and Section 3.2, ensuring that resampling remains unbiased requires careful tracking of each sample's supports, i.e., understanding which pixels could generate a particular sample. Without such tracking, it is extremely easy to under or overcount contributions in certain parts of the integration domains. This leads to bias in the form of unexpected brightening or darkening. Computing correct MIS weights is generally required for RIS or GRIS to remain unbiased if the candidate samples were produced from different sampling techniques (e.g., come from different pixels).

A problem of using the balance heuristic for MIS as described in Section 5.3 is the $O(M^2)$ cost grows quickly when M is large. One way to optimize performance is to use an incorrect, constant $1/M$ MIS weights (as in Bitterli et al.'s [1] biased implementation) and reduce the bias using neighbor rejection. In fact, neighbor rejection can then be thought of as an approximation of Veach's cutoff heuristic [31], where techniques with too low PDF values simply have their terms discarded in the MIS weight (neighbor rejection presumes that samples from a pair of incompatible domains have low importance on each other's domain).

Besides the biased approximate "MIS weights" offered by neighbor rejection, there are cheap, correct MIS weights we can use to make the estimator fully unbiased.

7.1.2 Contribution MIS weights

Bitterli et al. [1] show that it is possible to only evaluate the MIS weight for the selected sample and stay unbiased. This is called a *contribution MIS* in the GRIS framework [4].

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

For M-sample GRIS with resampling weights $w_i = m_i(T_i(X_i))\hat{p}(T_i(X_i))W_i \cdot \left|\frac{\partial T_i}{\partial X_i}\right|$, denote the selected index as s, it has been shown [4] that the selected sample $Y = T_s(X_s)$ can use the following unbiased contribution weight:

$$W_Y = \left[\frac{c_s(Y)}{m_s(Y)}\right]\frac{1}{\hat{p}(Y)}\sum_{j=1}^{M} w_j \tag{7.1}$$

as long as

$$\sum_{\substack{i=1 \\ y\in T_i(\operatorname{supp} X_i)}}^{M} c_i(y) = 1. \tag{7.2}$$

Each m_i can pretty much be an arbitrary function as long as it guarantees that $w_i > 0$ iff $X_i \in \mathcal{D}(T_i)$ and $\hat{p}(Y_i) > 0$. If m_i satisfies the same equation (Equation 7.2) as c_i, it is a proper resampling MIS weight and cancels c_i in

Equation 7.1, yielding the familiar unbiased contribution weight equation (Equation 3.2).

Bitterli et al. [1] use constant $m_i(x) = \frac{1}{M}$ and (generalized) balance heuristic for c_i. Since c_i is not in w_i and only c_s (c_i of the selected sample) needs to be evaluated, the cost is reduced to $O(M)$. Although debiasing with only contribution MIS weights performs reasonably well in direct lighting, it can add excessive noise to a selected sample's contribution if the difference between domains is large, which is especially noticeable in participating media [3]. In addition, convergence of sample distribution to the target PDF can only be achieved with proper resampling MIS weights m_i [4].

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[3]: Lin et al. (2021), 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

7.1.3 Pairwise MIS weights

To obtain a cheap resampling MIS weight, Bitterli [7] proposes pairwise MIS. A common assumption for multiple importance sampling is that the developer has no advance knowledge about which estimator might be *better*; each estimator may have places it ends up superior to others, but over the whole integration domain no clear winner exists.

[7]: Bitterli (2022), 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'

Instead, pairwise MIS considers a setting with M sampling techniques where one of them is *canonical*—this estimator covers the entire integration domain and produces relatively high-quality samples compared to other techniques. As an example, note that the canonical technique corresponds to the "current pixel" in spatial resampling and non-canonical techniques correspond to the neighboring pixels.

The core idea of pairwise MIS is to compute a "pairwise" balance heuristic MIS weight between pairs of samples: each sample pairs with the canonical sample (the sample produced by the canonical technique). This yields the following MIS weights (the canonical technique is assigned index c):

$$\begin{aligned} m_i(x) &= \frac{1}{M-1}\frac{p_i(x)}{p_i(x)+p_c(x)} && (i \neq c) \\ m_c(x) &= \frac{1}{M-1}\sum_{j\neq c}^{M}\frac{p_c(x)}{p_j(x)+p_c(x)} \,. \end{aligned} \tag{7.3}$$

It is easy to verify that this set of weights satisfies the requirements of valid MIS weights. Note that pairing with the canonical technique allows the MIS weight to account for how each technique compares to the canonical technique.

However, Equation 7.3 assigns disproportionally large weight to the canonical sample. To see why, assume that all sampling techniques are identical, i.e. $p_i(x) = p_j(x)$ for all i, j, $m_c(x)$ will be $M-1$ times larger than all other $m_i(x)$. To correct this, it is important to downweight $p_c(x)$. By requiring that the MIS weights are the same with identical techniques, it can be solved that the weighting factor for $p_c(x)$ needs to be $1/(M-1)$. This gives the modified equation:

$$
\begin{aligned}
m_i(x) &= \frac{1}{M-1}\frac{p_i(x)}{p_i(x)+p_c(x)/(M-1)} && (i \neq c)\\
m_c(x) &= \frac{1}{M-1}\sum_{j\neq c}^{M}\frac{p_c(x)/(M-1)}{p_j(x)+p_c(x)/(M-1)}\,.
\end{aligned}
\tag{7.4}
$$

Like in MIS using the balance heuristic, $\hat{p}$ can be used as proxy PDFs in the ReSTIR scenario, yielding the generalized pairwise MIS:

$$
\begin{aligned}
m_i(x) &= \frac{1}{M-1}\frac{\hat{p}_i(x)}{\hat{p}_i(x)+\hat{p}_c(x)/(M-1)} && (i \neq c)\\
m_c(x) &= \frac{1}{M-1}\sum_{j\neq c}^{M}\frac{\hat{p}_c(x)/(M-1)}{\hat{p}_j(x)+\hat{p}_c(x)/(M-1)}\,.
\end{aligned}
\tag{7.5}
$$

Because $\hat{p}_i$ are approximations, it is possible that $m_i(x)$, for $i \neq c$, are large even if samples are poor; this gives the canonical sample too low of a weight. To protect against this effect, pairwise MIS [7] can be designed to give the canonical sample a defensive constant in the weight by adding 1 to the sum. This gives the defensive form of pairwise MIS:

[7]: Bitterli (2022), 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'

$$
\begin{aligned}
m_i(x) &= \frac{1}{M}\frac{\hat{p}_i(x)}{\hat{p}_i(x)+\hat{p}_c(x)/(M-1)} && (i \neq c)\\
m_c(x) &= \frac{1}{M}\left(1+\sum_{j\neq c}^{M}\frac{\hat{p}_c(x)/(M-1)}{\hat{p}_j(x)+\hat{p}_c(x)/(M-1)}\right)\,.
\end{aligned}
\tag{7.6}
$$

Similar to balance heuristic (Equation 5.11), pairwise MIS weights have forms using confidence weights, rather than explicit sample counts based on M. Enabling confidence weights and shift mappings, using the shortcuts $\hat{p}_{\leftarrow i}$ (Equation 5.9), the non-defensive form (Equation 7.5) generalizes to

$$
\begin{aligned}
m_i(y) &= \frac{c_i\hat{p}_{\leftarrow i}(y)}{\left(\sum_{k\neq c}^{M}c_k\right)\hat{p}_{\leftarrow i}(y)+c_c\hat{p}_c(y)} && (i \neq c)\\
m_c(y) &= \sum_{j\neq c}^{M}\left(\frac{c_j}{\sum_{k\neq c}^{M}c_k}\right)\frac{c_c\hat{p}_c(y)}{\left(\sum_{k\neq c}^{M}c_k\right)\hat{p}_{\leftarrow j}(y)+c_c\hat{p}_c(y)}\,,
\end{aligned}
\tag{7.7}
$$

Tip 7.1 If $\hat{p}_{\leftarrow i} = \hat{p}_{\leftarrow j} = \hat{p}_c$ for all i, j, non-defensive pairwise MIS reduces to

$$m_i = \frac{c_i}{\sum_{k=1}^{M}c_k} \quad \text{for all } i.$$

and the defensive form (Equation 7.6) generalizes to

$$
\begin{aligned}
m_i(y) &= \left(\frac{\sum_{k\neq c}^{M}c_k}{\sum_{k=1}^{M}c_k}\right)\frac{c_i\hat{p}_{\leftarrow i}(y)}{\left(\sum_{k\neq c}^{M}c_k\right)\hat{p}_{\leftarrow i}(y)+c_c\hat{p}_c(y)} && (i \neq c)\\
m_c(y) &= \frac{c_c}{\sum_{k=1}^{M}c_k}+\sum_{j\neq c}^{M}\left(\frac{c_j}{\sum_{k=1}^{M}c_k}\right)\frac{c_c\hat{p}_c(y)}{\left(\sum_{k\neq c}^{M}c_k\right)\hat{p}_{\leftarrow j}(y)+c_c\hat{p}_c(y)}\,.
\end{aligned}
\tag{7.8}
$$

Tip 7.2 Defensive pairwise MIS lerps between non-defensive and giving all weight to the canonical sample with $t_c = c_c/\sum_k c_k$. E.g., if $\hat{p}_{\leftarrow i} = \hat{p}_{\leftarrow j} = \hat{p}_c$ for all i, j, it reduces to

$$
\begin{aligned}
m_i &= (1-t_c)\cdot\frac{c_i}{\sum_{k=1}^{M}c_k} \quad (i \neq c)\\
m_c &= t_c+(1-t_c)\cdot\frac{c_c}{\sum_{k=1}^{M}c_k}\,.
\end{aligned}
$$

ReSTIR PT [4] observes the $O(M)$ pairwise MIS gives comparable convergence behavior as the $O(M^2)$ balance heuristic and adopts the defensive

[4]: Lin et al. (2022), 'Generalized Resampled Importance Sampling'

Algorithm 7: Generalized Pairwise MIS (Defensive Variant)

Input : $y \in \Omega$ where $m_i(y)$ is evaluated.
Input : Original $x \in \Omega_i$ that yielded y. We assume $y = T_i(x)$.
Input : Jacobian determinant $\left|T_i'(x)\right|$ of the shift $y = T_i(x)$.
Output: Generalized defensive pairwise MIS weight $m_i(y)$.

function pHatFrom(j, y) // $\hat{p}_{\leftarrow j}(y)$ for generic $y \in \Omega$.
  $x_j, Jx_j \leftarrow T_j^{-1}(y), |T_j^{-1\prime}(y)|$ // Shift y into Ω_j and eval Jacobian.
  if $x_j \neq \emptyset$ **then** // If shift succeeded
    return $\hat{p}_j(x_j) \cdot Jx_j$ // $\hat{p}_j(T_j^{-1}(y))\ |T_j^{-1\prime}(y)|$
  return 0

function pHatFrom_opt$(j, x, |T_j'(x)|)$ // $\hat{p}_{\leftarrow j}(y)$ optimized for $y = T_j(x)$
  return $\hat{p}_j(x)/|T_j'(x)|$ // $\hat{p}_j(T_j^{-1}(y))\ |T_j^{-1\prime}(y)|$.

function GenPairwiseMIS_canonical(y) // $m_c(y)$
  $c_{\text{tot}} \leftarrow \sum_{k=1}^{M} c_k$
  $m_c \leftarrow \frac{c_c}{c_{\text{tot}}}$
  $m_{\text{num}} \leftarrow c_c \cdot \hat{p}_c(y)$
  for $j \leftarrow 1$ **to** M; $j \neq c$ **do** // $\sum_{j\neq c}^{M} \cdots$
    $m_{\text{den}} \leftarrow m_{\text{num}} + (c_{\text{tot}} - c_c) \cdot \text{pHatFrom}(j, y)$ // $m_{\text{num}} + (\sum_{k\neq c}^{M} c_k)\hat{p}_{\leftarrow j}(y)$
    $m_c \leftarrow m_c + \frac{c_j}{c_{\text{tot}}} \cdot \frac{m_{\text{num}}}{m_{\text{den}}}$
  return m_c // $\frac{c_c}{\sum_{k=1}^{M} c_k} + \sum_{j\neq c}^{M} \left(\frac{c_j}{\sum_{k=1}^{M} c_k}\right) \frac{c_c \hat{p}_c(y)}{\left(\sum_{k\neq c}^{M} c_k\right)\hat{p}_{\leftarrow j}(y) + c_c \hat{p}_c(y)}$

function GenPairwiseMIS_noncanonical$(i, y; x, \left|T_i'(x)\right|)$ // $m_i(y)$ $(y = T_i(x))$
  $c_{\text{tot}} \leftarrow \sum_{k=1}^{M} c_k$
  $m_{\text{num}} = (c_{\text{tot}} - c_c) \cdot \text{pHatFrom\_opt}(i, x, |T_i'(x)|)$ // $(\sum_{k\neq c}^{M} c_k)\, \hat{p}_{\leftarrow i}(y)$
  $m_{\text{den}} = m_{\text{num}} + c_c \cdot \hat{p}_c(y)$ // $(\sum_{k\neq c}^{M} c_k)\, \hat{p}_{\leftarrow i}(y) + c_c \hat{p}_c(y)$
  return $\frac{c_i}{c_{\text{tot}}} \cdot \frac{m_{\text{num}}}{m_{\text{den}}}$ // $\frac{\sum_{k\neq c}^{M} c_k}{\sum_{k=1}^{M} c_k} \cdot \frac{c_i\, \hat{p}_{\leftarrow i}(y)}{(\sum_{k\neq c}^{M} c_k)\, \hat{p}_{\leftarrow i}(y) + c_c \hat{p}_c(y)}$

Note:
Return 0 if y is a null sample; this avoids $0/0$. See Tip 3.13.

form as the default choice for spatial resampling in GRIS (see Algorithm 7 for pseudocode). A generalized family of pairwise MIS weights is also discussed in Lin et al. [4].

7.1.4 Biased MIS Weights

With an understanding how the bias arises, careful algorithmic modifications can compute slightly incorrect MIS weights (on purpose) in the name of efficiency.

Imagine reuse between pixels i and j, which have selected samples X_i and X_j. Using the balance heuristic for MIS weights requires evaluating either:

$$m_i(X_i) = \frac{p_i(X_i)}{p_i(X_i) + p_j(X_i)} \qquad \text{or} \qquad m_j(X_j) = \frac{p_j(X_j)}{p_i(X_j) + p_j(X_j)}. \tag{7.9}$$

While $p_i(X_i)$ and $p_j(X_j)$ are already computed as part of resampling, $p_i(X_j)$ and $p_j(X_i)$ require reevaluating a sample in a pixel that *did not* generate it. This is new computation. Very concretely, if our samples X are rays or paths, reevaluating them at a neighbor requires *tracing new rays*! This can be expensive or simply add complex new engineering to make it possible.

For example, when doing temporal reuse where pixel j comes from the previous frame, then $p_j(X_i)$ requires taking candidate X_i, generated during the current frame i, and reevaluating its path using the *previous frame's* data. This includes using the prior frame BVH ray acceleration structure, which is obviously unappealing.

Fortunately, after understanding the causes of bias, you can make informed decisions on whether biased approaches are objectionable. For instance, using the current frame BVH as a stand in for the prior frame BVH; assuming $p_j(X_i) = 0$; or recomputing $p_j(X_i)$ using last frame's data but assuming visibility does not change.

In particular, consider the simple balance heuristic MIS weight from Equation 7.9:

$$m_i(X_i) = \frac{p_i(X_i)}{p_i(X_i) + \boxed{p_j(X_i)}} \tag{7.10}$$

The boxed weight $p_j(X_i)$ is the tricky one, requiring more expensive computations using last frame's data. But you can consider the expected bias in very simplistic terms. If you replace $p_j(X_i)$ by some biased approximation $\tilde{p}_j(X_i)$, either:

- $\tilde{p}_j(X_i) > p_j(X_i)$, lowering $m_i(X_i)$ and adding a darkening bias;
- $\tilde{p}_j(X_i) = p_j(X_i)$, adding no bias for X_i; or
- $\tilde{p}_j(X_i) < p_j(X_i)$, increasing $m_i(X_i)$ and adding a brightening bias.

And obviously, approximations $\tilde{p}_j$ are not limited to always be greater than or less than the correct probability density. For some X_i, $\tilde{p}_j > p_j$ and for others $\tilde{p}_j < p_j$. This can cause both darkening and brightening biases in different parts the image or under different type of motion or animation.

As a quick example, assuming $p_j(X_i) = 0$ means any sample generated this frame *could not* have been selected last frame. This is clearly not (always) true, especially in static scenes. Some such samples were likely selected last frame and forwarded via spatiotemporal reuse to the current frame. Because of this simplistic assumption, these samples are over-represented in our sample pool; this means we overcount, giving a brightening bias.

7.2 Low-level optimization

For low-level optimization, defining specific performance goals can be vital to achieving a target budget. These performance goals usually need to consider the computation model of the hardware. Various metrics you could reasonably optimize exist, including:

- Minimize the per-pixel shadow ray count (Bitterli et al.'s [1] original goal, targeting scenes with millions of lights).
- Minimize the number of *paths* traced.
- Maximize sample *reuse*; path samples are costly, so reuse each as much as possible to minimize cost per reuse.
- Minimize correlation in final shading, so denoisers behave better.
- Maximize parallelization and streaming reuse for GPU utilization (e.g., using weighted reservoir sampling [30]).
- Minimize size of intermediate buffers (e.g., reservoir size).
- Minimize memory bandwidth.
- Minimize execution divergence (ensuring maximal thread counts active in each GPU warp).
- Minimize memory divergence (to avoid thrashing caches and minimizing memory access costs).
- Minimize frame time. (ReSTIR benefits significantly from *temporal* reuse, so overall quality may improve by reducing the quality gained *per-frame* if you can instead reuse across frames much more quickly.)
- Plus other traditional low-level optimization targets, e.g., minimizing register usage.

[1]: Bitterli et al. (2020), 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'

[30]: Chao (1982), 'A General Purpose Unequal Probability Sampling Plan'

Additionally, some optimization techniques remain unbiased, while others fundamentally add bias. Others add bias unless you apply more sophisticated mathematics; this may not be a desirable trade for your application.

An obvious question that occurs to *every* experienced rendering engineer considering ReSTIR is, "why not sparsely sample on some world-space grid to improve performance?" (e.g., [8, 20]). This reduces ray (or path) count and total reservoir memory in a relatively linear way, but ReSTIR memory usage is already fairly minimal and ray count is only vital on very low-end ray tracing hardware. But in exchange, grid based reuse adds bias that we are only just learning to control (with as-yet unpublished theory). Other optimizations might give you more well-rounded performance improvements with less bias baggage. Here we provide three examples of low-level optimizations in ReSTIR DI and ReSTIR PT.

[8]: Boksansky et al. (2021), 'Rendering Many Lights with Grid-Based Reservoirs'
[20]: Boissé (2021), 'World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination'

7.2.1 Sample tiling in ReSTIR DI

When we first started optimizing Bitterli et al.'s [1] ReSTIR for direct lighting, we asked folks on our performance team for advice. Our Amusement Park has over 3 million emissive triangles and just picking random light candidates took up to 25 milliseconds, without any fancy spatiotemporal reuse!

Figure 7.2: Amusement Park with over 3 million emissive triangle lights.

One fact obvious from any basic profiling was: as lighting complexity increased, so did our memory access costs. Essentially, we were thrashing our memory caches. Each random light sample selected a light residing in a different cache line. The first, and only somewhat facetious, response from our performance analysis team was, "design a different algorithm, without random sampling!"

But what they really meant was "redesign the algorithm to avoid accessing multiple random cache lines every pixel," or more simply "amortize the memory access costs." So... how to do this?

The great thing about the Amusement Park scene is its over-the-top lighting complexity helps identify places where naive algorithms are wasteful. In a 1920×1080 image, we have around 2 million pixels. If each pixel selects one light (via ReSTIR) to shade, we *at most* use two thirds of the lights for shading each frame! We could clearly reduce cache thrash if we reorganize (and compact) the lights each frame so we only search those relevant for that frame's shading.

But determining which lights are relevant to each frame is non-obvious. Perhaps we should start simpler, maybe with a stratification approach?

What if we used only a quarter of the lights in the scene? If we always picked from this same small subset of lights, the other 3/4 would never appear to emit light. But if frames alternate which 1/4 of the lights are sampled, over time we could pick from any light. And because ReSTIR reuses samples spatiotemporally, very important lights selected two or three frames ago can still impact lighting this frame.

But this can be amplified to dramatically reduce memory access costs.

While sampling from a rotating subset of only 1/4 of the lights each frame seems reasonable, sampling from only 1/1000 of the lights is not obviously still useful.

But what if only considering sampling for a small image region, say a 16×16 pixel tile? If each pixel in that tile selected a random light via ReSTIR, that tile needs at most 256 unique lights. Perhaps we could pick *that* set of samples from a subset of 1024 or 2048 scene lights?

That is the basic idea behind sample tiling, giving the following simple algorithm:

1. Each frame, generate light subsets S, each containing a random selection of all scene lights. Select lights for subsets using the PDF p normally used to select candidates without sample tiling (e.g., pick lights proportional to their intensity).
2. For each screen tile (e.g., 8×8 or 16×16 pixels), pick one of this frame's light subsets S_i to use.
3. For each pixel in a screen tile, select the needed number of candidates from tile i's selected subset S_i. Pick from the lights in the tile uniformly randomly (i.e., probabilty of $1/N$ each, if each tile has N lights).

Usually, generating 128 light subsets each containing 1024 light samples is sufficient across a wide variety of scenes, including the Amusement Park [5]. However, for simple scenes with few lights (i.e., $\ll$ 128,000) the overhead of building these tiles (under 0.1 ms) may overwhelm any caching benefits.

[5]: Wyman et al. (2021), 'Rearchitecting Spatiotemporal Resampling for Production'

It turns out this precomputed sample tiling is a degenerate form of resampled importance sampling (see Wyman and Panteleev [5]), where the target function of this RIS step is p.

This degenerate sample tiling is a way of using RIS to *reorganize* sampling to be more cache coherent. This is an extremely powerful idea, and can be used in more complex sampling scenarios than direct lighting.

7.2.2 Lighting with many analytic light types

Many applications may have multiple light types, each with its own (potentially expensive) sampling code. For instance, you might have emissive spheres, quad, cylinders, triangles, environment maps, lines, points, spotlights, meshes, etc.

If each pixel selects a different analytic type to sample, you likely inject both execution divergence and cache thrashing into the per-pixel sampling code. For instance, one pixel might sample a sphere light while its neighbors sample a triangle and an environment map. If these three pixels are part of a single GPU warp, the three sampling procedures will likely happen serially (due to execution divergence) rather than in parallel.

By first creating buffers of presampled locations on each type of emissive, and then feeding these buffers as input to the sample tiling in Section 7.2.1, this execution divergence can be moved out of the inner loop.

Essentially, per pixel during the render loop, we sample from precomputed point lights that all have the same structure. Some of these came from sphere lights, some from triangles, some from environment maps, etc., but the potentially expensive, per-primitive sampling procedures happen once per frame in a coherent way.

7.2.3 Accelerating hybrid shift

Hybrid shift in ReSTIR PT performs random replay and vertex reconnection to reuse a path. This includes multiple tasks: tracing new subpath, testing visibility rays, and re-evaluating BSDFs. A naive implementation directly following Algorithm 7 usually results in inflated shader time. This has two main causes.

- A shader that contains multiple complex procedures that are dependent on each other or have a nested structure often very high register usage, lowering the warp occupancy, and potentially causes register spilling to inflate memory cost.
- Having large execution divergence across threads can lower the effective computation throughput. A small percentage of pixels having path tracing work can be as expensive as all pixels doing path tracing.

To tackle these problems, we recommend two optimizations:

1. Use smaller kernels instead of a big kernel: have a dedicated random replay kernel that only does path tracing and a dedicated reconnection kernel that performs BSDF re-evaluatation and visibility ray tests.

2. Perform stream compaction to map threads only to non-empty ray tracing tasks: since many path samples do not need random replay to reconnect, performing path tracing in a compact way avoids having idle threads running within the warp doing nothing.

But such optimizations have additional memory overhead: because the RIS step happens after the complete evaluation of the target function $\hat{p}$, intermediate results of random replay (containing partial path throughput) need to be written to the global memory in the end of the random replay shader and read from the global memory in the reconnection shader. But the additional memory overhead is usually small compared to overall reduced shader time. Using the Veach Ajar scene for example, with the first optimization, we have observed about 40% reduction in shader time related to spatiotemporal resampling. The second optimization further reduces 40% of shader time on top of the first optimization.

8 Experiences in game integration

Please see Pawel and Giovanni's slides from SIGGRAPH 2023, available on our course webpage, which discuss some of their key experiences and take-aways from integrating ReSTIR into CD Project Red's *Cyberpunk 2077* as part of its *RT Overdrive Mode* and recent *Phantom Liberty* expansion.

9 Advice for getting started

As authors of this course, we have all thought about resampling and ReSTIR for years. We've collectively written (and rewritten) code, prototypes, demos, SDKs, and integrated ReSTIR into more complex code bases.

You should start simple.

Probably every ReSTIR implementation around today has confusing bits you will not initially understand. This is akin to how usually everyone's first path tracer has "off by π" issues; as researchers and rendering engineers we're still wrapping our minds around how to best write this code in a clean and understandable way. Sometimes the first paper is not the right place to go, even if it's simpler to understand. (Sorry.)

So some advice from us to you, after helping out numerous researchers and engineers get up to speed on ReSTIR:

- *Start with a simple ground-truth Monte Carlo path tracer.* No need to have fancy importance sampling, but it needs to run in the same code, on the same scenes where you plan to use ReSTIR. You do not want to spend months debugging your ReSTIR implementation or integration only to discover, at the very end, that it is biased in some unacceptable way. (This has happened.) You want to discover this bias when you introduce it. Compare to ground truth. Frequently.
- *Start simple, with basic RIS.* Talbot's basic RIS [6] is fairly straightforward to implement without bias. It's pretty easy to understand. If RIS will not converge to ground truth, neither will your experiments with spatiotemporal reuse.
- *Think about rendering bias.* Most real-time engineers without an offline rendering background never worry about bias... after all, we always approximate in real-time rendering! Many of us felt the same way. Now, we have *all* concluded that managing bias is super important, even *in game*. With spatiotemporal sample reuse, bias spreads around the screen extremely quickly. And with multi-bounce paths, it shows up in extremely odd ways. Your art director may disapprove.
- *Spatial reuse alone is easier to debug; combining with temporal reuse gives better quality.* Moving beyond basic RIS, next add spatial reuse. Without scene changes between samples (as in temporal reuse), it is much easier to validate. Spatial reuse should give clearly visible improvement. But move on quickly after validating that spatial reuse works correctly, since interleaving spatial and temporal reuse improves quality much more significantly than spatial reuse alone.
- *Don't try to get too clever too fast.* If you grab RTXDI [9], there are a ton of options. Checkerboarding, sample permutations, boiling suppression, etc., etc. Many were never intended to be unbiased, and options may not have been tested in all permutations. Wait to try

clever techniques until you know the basics work (and you need the improvements those clever techniques provide).

- *Basic ReSTIR gives you probability distributions at a* ***point****.* Generally, a reservoir is *not* valid over, say, an entire voxel. You can store and use reservoirs that way, but it is very difficult to avoid adding bias (and magnifying correlations within the voxel).
- *Reuse visibility very carefully.* An original appeal of ReSTIR was the ability to reduce ray budgets by reusing visibility samples. Visibility reuse also causes many problematic biases people have great difficulty debugging. (Arguably, it causes *most* difficult-to-debug biases.) Consider always using visibility in your target functions and MIS weights until you have validated your code works with full visibility. Only then accelerate your algorithm by incrementally starting to reuse ray queries.
- *ReSTIR accelerates in multiple ways.* One is by *amortizing* sample costs across pixels. This benefit remains, even when not reusing visibility.
- *Think a bit about ReSTIR as subsampling the integration domain.* If doing environment lighting using a light probe, an obvious way to gain performance is to coarsen the probe texture. Now, if your integration domain isn't a hemisphere of incident colors, but rather a high-dimensional path space, how do you "coarsen" that domain? Perhaps you could *reuse* samples rather than tracing new independent ones all the time?

Bibliography

Here are the references in citation order.

[1] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 'Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). DOI: 10/gg8xc7. (Visited on 08/23/2020) (cited on pages i, ii, 1, 7, 15, 20, 38, 39, 43).

[2] Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 'ReSTIR GI: Path Resampling for Real-Time Path Tracing'. In: *Computer Graphics Forum* 40.8 (2021), pp. 17–29. DOI: 10/gqwmdx (cited on pages i, ii, 27, 28).

[3] Daqi Lin, Chris Wyman, and Cem Yuksel. 'Fast Volume Rendering with Spatiotemporal Reservoir Resampling'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40.6 (Dec. 10, 2021), 279:1–279:18. DOI: 10/grrjd6 (cited on pages i, ii, 27, 34, 35, 39).

[4] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 'Generalized Resampled Importance Sampling: Foundations of ReSTIR'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 22, 2022), 75:1–75:23. DOI: 10/gqjn7b. (Visited on 07/23/2022) (cited on pages i, ii, 7, 9, 12, 14, 19, 22, 23, 25, 27–30, 38–41).

[5] Chris Wyman and Alexey Panteleev. 'Rearchitecting Spatiotemporal Resampling for Production'. In: *High-Performance Graphics - Symposium Papers*. Eurographics Association, 2021. DOI: 10/grrjkk (cited on pages i, ii, 20, 44).

[6] Justin F. Talbot, David Cline, and Parris Egbert. 'Importance Resampling for Global Illumination'. In: *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*. Eurographics Association, June 2005, pp. 139–146. DOI: 10/gfzsm2 (cited on pages i, 1, 3, 7, 8, 11, 18, 48).

[7] Benedikt Bitterli. 'Correlations and Reuse for Fast and Accurate Physically Based Light Transport'. PhD thesis. Hanover, NH: Dartmouth College, Jan. 1, 2022 (cited on pages ii, 12, 21, 39, 40).

[8] Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 'Rendering Many Lights with Grid-Based Reservoirs'. In: *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX*. Ed. by Adam Marrs, Peter Shirley, and Ingo Wald. Berkeley, CA: Apress, 2021, pp. 351–365. DOI: 10.1007/978-1-4842-7185-8_23. (Visited on 02/10/2023) (cited on pages ii, 43).

[9] NVIDIA. *NVIDIA® RTX Direct Illumination*. 2021. URL: https://developer.nvidia.com/rtxdi (visited on 05/28/2021) (cited on pages ii, 48).

[10] Alex Battaglia. *Sword and Fairy 7 is the cutting-edge PC exclusive nobody's talking about*. 2021. URL: https://www.eurogamer.net/digitalfoundry-2021-sword-and-fairy-7-tech-review (visited on 02/07/2023) (cited on page ii).

[11] Andrew Burnes. *Portal with RTX Out Now: A Breathtaking Reimagining Of Valve's Classic With Full Ray Tracing & DLSS 3*. 2022. URL: https://www.nvidia.com/en-us/geforce/news/portal-with-rtx-ray-tracing/ (visited on 02/07/2023) (cited on page ii).

[12] Jiayin Cao. *Understanding The Math Behind ReSTIR DI*. 2022. URL: https://agraphicsguynotes.com/posts/understanding_the_math_behind_restir_di/ (visited on 02/07/2023) (cited on page ii).

[13] Julien Guertault. *Reading list on ReSTIR*. 2022. URL: https://lousodrome.net/blog/light/2022/05/14/reading-list-on-restir/ (visited on 02/07/2023) (cited on page ii).

[14] Shubham Sachdeva. *Spatiotemporal Reservoir Resampling (ReSTIR) - Theory and Basic Implementation*. 2021. URL: https://gamehacker1999.github.io/posts/restir/ (visited on 02/07/2023) (cited on page ii).

[15] Jacco Bikker. *Lecture 14 - "TAA & ReSTIR"*. 2023. URL: http://www.cs.uu.nl/docs/vakken/magr/2022-2023/slides/lecture%2014%20-%20ReSTIR.pdf (visited on 02/07/2023) (cited on page ii).

[16] Tomasz Stachowiak. *Global Illumination in 'kajiya' Renderer*. 2022. URL: https://github.com/EmbarkStudios/kajiya/blob/main/docs/gi-overview.md (visited on 02/07/2023) (cited on page ii).

[17] Mr. Zyanide. *Shared Twitter results for Jedi Outcast integration*. 2023. URL: https://twitter.com/MZyanide/status/1610172199146586112 (visited on 02/07/2023) (cited on page ii).

[18] Adam Badke. 'Next event estimation via reservoir-based spatio-temporal importance resampling'. MA thesis. Simon Fraser University, June 2021 (cited on page ii).

[19] Ege Ciklabakkal, Adrien Gruson, Iliyan Georgiev, Derek Nowrouzezahrai, and Toshiya Hachisuka. 'Single-Pass Stratified Importance Resampling'. In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 41.4 (2022). DOI: 10.1111/cgf.14585. (Visited on 07/21/2022) (cited on page ii).

[20] Guillaume Boissé. 'World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination'. In: *SIGGRAPH Asia 2021 Technical Communications*. New York, NY, USA: ACM Press, Dec. 14, 2021, pp. 1–4. DOI: 10/grrjbg (cited on pages ii, 43).

[21] Xander Hermans. 'The Effectiveness of the ReSTIR Technique When Ray Tracing a Voxel World'. MA thesis. Utrecht University, July 2022. (Visited on 02/10/2023) (cited on page ii).

[22] Fuyan Liu and Junwen Gan. 'Light Subpath Reservoir for Interactive Ray-Traced Global Illumination'. In: *Computers & Graphics* 111 (Apr. 1, 2023), pp. 37–46. DOI: 10/grrjgw (cited on page ii).

[23] Shinji Ogaki. 'Vectorized Reservoir Sampling'. In: *SIGGRAPH Asia 2021 Technical Communications*. New York, NY, USA: ACM Press, Dec. 14, 2021, pp. 1–4. DOI: 10/grrjhq. (Visited on 02/09/2023) (cited on page ii).

[24] Stefan Krake. *hdRstr: A ReSTIR/RTXDI-based Hydra Render Delegate*. 2021. URL: https://stkrake.net/ (visited on 02/07/2023) (cited on page ii).

[25] Stefan Krake. *blRstr: A ReSTIR/RTXDI-based Blender Render Engine*. 2022. URL: https://stkrake.net/blRstr/ (visited on 02/07/2023) (cited on page ii).

[26] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 'Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination'. In: *Proceedings of High Performance Graphics*. New York, NY, USA: ACM, 2017, 2:1–2:12. DOI: 10/ggd8dg. (Visited on 12/10/2019) (cited on page 1).

[27] Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 'Gradient Estimation for Real-Time Adaptive Temporal Filtering'. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.2 (Aug. 2018), 24:1–24:16. DOI: 10/ggd8dh. (Visited on 12/10/2019) (cited on page 1).

[28] Takafumi Saito and Tokiichiro Takahashi. 'Comprehensible Rendering of 3-D Shapes'. In: *Computer Graphics (Proceedings of SIGGRAPH)* 24.4 (Sept. 1990), pp. 197–206. DOI: 10/fp3t53 (cited on page 1).

[29] Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 'Path Guiding in Production'. In: *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. Los Angeles, California, 2019. DOI: 10.1145/3305366.3328091 (cited on page 1).

[30] Min-Te Chao. 'A General Purpose Unequal Probability Sampling Plan'. In: *Biometrika* 69.3 (Dec. 1, 1982), pp. 653–656. DOI: 10/fd87zs (cited on pages 1, 15, 43).

[31] Eric Veach. 'Robust Monte Carlo Methods for Light Transport Simulation'. PhD thesis. Stanford University, Dec. 1997 (cited on pages 2, 38).

[32] Eric Veach and Leonidas J. Guibas. 'Optimally Combining Sampling Techniques for Monte Carlo Rendering'. In: *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 29. ACM Press, Aug. 1995, pp. 419–428. DOI: 10/d7b6n4 (cited on pages 6, 12, 13, 37).

[33] Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 'Optimal Multiple Importance Sampling'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 38.4 (July 2019), 37:1–37:14. DOI: 10/gf5jbj. (Visited on 07/29/2019) (cited on page 7).

[34] E. R. Woodcock, T. Murphy, P. J. Hemmings, and T. C. Longworth. 'Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry'. In: *Applications of Computing Methods to Reactor Problems*. Argonne National Laboratory. 1965 (cited on pages 7, 35).

[35] Henrik Wann Jensen. 'Global Illumination Using Photon Maps'. In: *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. Vienna: Springer-Verlag, June 1996, pp. 21–30. DOI: 10/fzc6t9 (cited on page 7).

[36] Hao Qin, Xin Sun, Qiming Hou, Baining Guo, and Kun Zhou. 'Unbiased Photon Gathering for Light Transport Simulation'. In: *ACM Transactions on Graphics* 34.6 (Oct. 26, 2015). DOI: 10/f7wrc6 (cited on page 7).

[37] Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 'Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). DOI: 10/gg8xc8. (Visited on 08/23/2020) (cited on page 7).

[38] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 'Gradient-Domain Metropolis Light Transport'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32.4 (July 2013), 95:1–95:12. DOI: 10/gbdghd (cited on pages 22, 29).

[39] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 'Gradient-Domain Path Tracing'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 34.4 (July 27, 2015), p. 123. DOI: 10/gfzrhn (cited on pages 22, 29, 30, 32).

[40] Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 'A Survey on Gradient-Domain Rendering'. In: *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 455–472 (cited on pages 22, 23, 29, 30).

[41] Yusuke Tokuyoshi. 'Efficient Spatial Resampling Using the PDF Similarity'. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.1 (2023), pp. 1–19 (cited on page 28).

[42] Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. 'Improved Sampling for Gradient-Domain Metropolis Light Transport'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 33.6 (2014). DOI: 10/f6r2hp (cited on pages 29, 30).

[43] Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 'The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014), 102:1–102:13. DOI: 10/f6cz85 (cited on page 29).

[44] Wenzel Jakob and Steve Marschner. 'Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 31.4 (July 2012), 58:1–58:13. DOI: 10/gfzq4p (cited on page 29).

Abbreviations

GPU
graphics processing unit. 1

MC
Monte Carlo. 4–7

MIS
multiple importance sampling. vi, 6, 8–14, 28, 32, 33, 37–42

NEE
next-event estimation. 27, 28, 31–33

PDF
probability density function. v, 1, 2, 4–14, 38

ReSTIR
reservoir-based spatiotemporal importance resampling. vi, 1, 4–6, 9, 37–46

RIS
resampled importance sampling. v, 1, 3, 4, 7–14, 37, 38, 44–46

RNG
random number generator. 33

WRS
weighted reservoir sampling. 15

Symbols

Page List	Symbol	Description	Notation
6, 9	$m(X)$	The MIS weight of the random variable X.	MIS weight
8, 9	w_i	The probability of selecting candidate i from the list of candidates $(X_1, \ldots, X_M)$.	resampling weight
22	T	A shift mapping.	shift mapping
5	$\text{supp}(X), \text{supp}(f)$	The support of a random variable X or a function f.	support
9	$\hat{p}(x)$	The target function at x.	target function
9	$\bar{p}(x)$	The target PDF at x.	target PDF
7–9	W_X	The unbiased contribution weight of X. If X has known PDF $p(X)$, use $W_X = \frac{1}{p(x)}$.	unbiased contribution weight