Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting

BENEDIKT BITTERLI, Dartmouth College
CHRIS WYMAN, NVIDIA
MATT PHARR, NVIDIA
PETER SHIRLEY, NVIDIA
AARON LEFOHN, NVIDIA
WOJCIECH JAROSZ, Dartmouth College

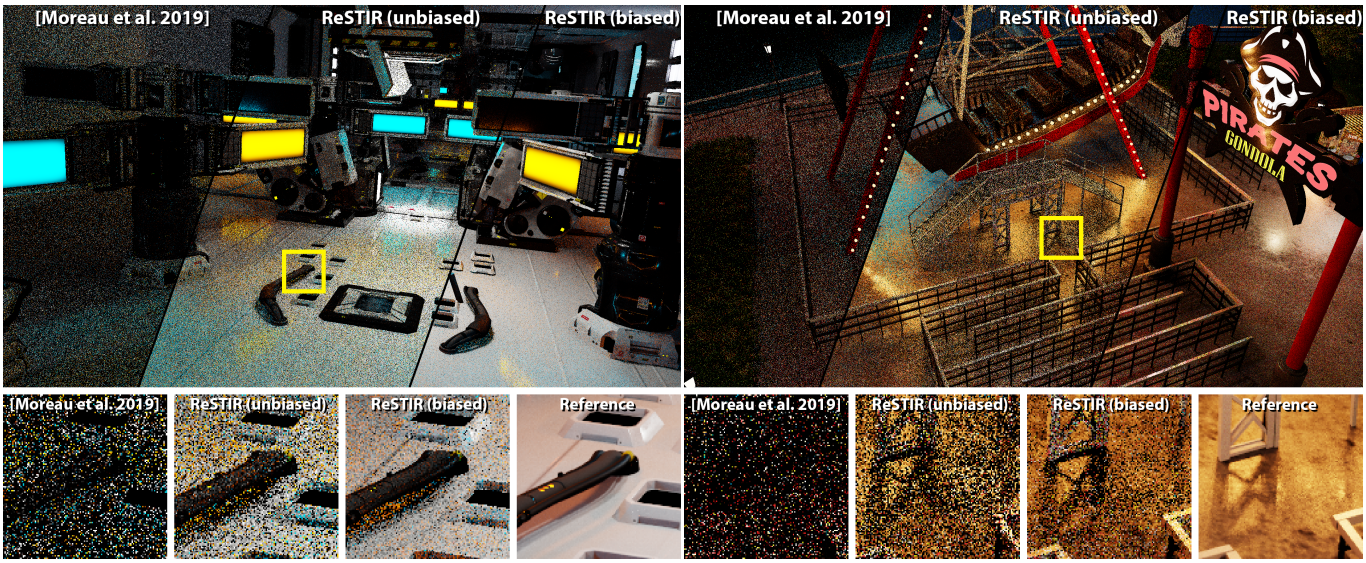


Fig. 1. Two complex scenes ray traced with direct lighting from many dynamic lights. (Left) A still from the Zero Day video [Winkelmann 2015] with 11,000 dynamic emissive triangles. (Right) A view of one ride in an Amusement Park scene containing 3.4 million dynamic emissive triangles. Both images show three methods running in equal time on a modern GPU, from left to right: Moreau et al. [2019]'s efficient light-sampling BVH, our new unbiased estimator, and our new biased estimator. The Zero Day image is rendered in 15 ms and Amusement Park in 50 ms, both at 1920 × 1080 resolution. Zero Day ©beeple, Pirate Ship ©sema edis

Efficiently rendering direct lighting from millions of dynamic light sources using Monte Carlo integration remains a challenging problem, even for off-line rendering systems. We introduce a new algorithm—ReSTIR—that renders such lighting interactively, at high quality, and without needing to maintain complex data structures. We repeatedly resample a set of candidate light samples and apply further spatial and temporal resampling to leverage information from relevant nearby samples. We derive an unbiased Monte Carlo estimator for this approach, and show that it achieves equal-error 6×-60× faster than state-of-the-art methods. A biased estimator reduces noise further and is 35×-65× faster, at the cost of some energy loss. We implemented our approach on the GPU, rendering complex scenes containing up to 3.4 million dynamic, emissive triangles in under 50 ms per frame while tracing at most 8 rays per pixel.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Photorealistic rendering, resampled importance sampling, real-time rendering, reservoir sampling

ACM Reference Format:
Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 1 (July 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

Authors' addresses: Benedikt Bitterli, Dartmouth College, benedikt.m.bitterli.gr@dartmouth.edu; Chris Wyman, NVIDIA, chris.wyman@acm.org; Matt Pharr, NVIDIA, matt.pharr@gmail.com; Peter Shirley, NVIDIA, ptrshrl@gmail.com; Aaron Lefohn, NVIDIA, 2788 San Tomas Expressway, Santa Clara, CA, 95051, alefohn@nvidia.com; Wojciech Jarosz, Dartmouth College, Department of Computer Science, 9 Maynard St. Hanover, NH, 03755, wojciech.k.jarosz@dartmouth.edu.

0730-0301/2020/7-ART1 $15.00
https://doi.org/10.1145/3386569.3392481

Fig. 2. While existing denoisers (e.g., Chaitanya et al. [2017]; NVIDIA Research [2017]; Schied et al. [2018]) vastly improve image quality at a given sampling rate, they cannot reconstruct features that are missing from their input samples. Our work improves the sampling quality at a given computation budget, enabling existing denoisers to produce better results. Here we show Moreau et al. [2019]'s light BVH, our unbiased (Section 4) and biased (Section 3) methods with and without the OptiX denoiser [NVIDIA Research 2017]. The Amusement Park's carousel image is rendered in 42 ms at 1920 × 1080 resolution (without denoising) with 3.4 million animated lights. Carousel ©carousel_world

1 INTRODUCTION

In recent years, Monte Carlo path tracing has been widely adopted for offline rendering [Christensen and Jarosz 2016; Fascione et al. 2017] and is seeing increasing use in real-time applications [Schied 2019] with the arrival of specialized hardware support for ray intersection tests [Parker et al. 2010; Wyman et al. 2018]. Even in offline rendering, without the constraints of real-time, direct lighting with many emissive objects remains challenging; it's not feasible to trace shadow rays to all of the lights, and finding the lights that contribute most at a given point depends on each light's visibility to that point, the distribution of the scattering function (BSDF or phase function) at the point, and the light source's power and emissive characteristics.

Real-time rendering adds even more challenges: the scenes to be rendered are dynamic and the renderer generally has no future knowledge of how the scene will change, as that may be affected by user interaction. Furthermore, only a few rays can currently be traced at each pixel, so finding important lights is even more critical, yet there is a limited amount of time to build and update data structures to aid light sampling [Moreau et al. 2019]. This is true even for the restricted case of direct lighting at the first camera vertex, which we consider in this paper.

These constraints have spurred research in denoising and reconstructing images from noisy low-sample-per-pixel rendered images. While great strides have been made in this area in both offline [Vogels et al. 2018] and real-time [Schied et al. 2018] rendering, a limited amount of processing time is available for real-time denoisers since time spent filtering takes away from the available frame time. Denoising is particularly challenging with low sample-count images; as shown in Fig. 2, improving the quality of samples provided to a denoiser can significantly increase its effectiveness.

We introduce a method to sample one-bounce direct lighting from many lights that is suited to real-time ray tracing with fully dynamic scenes (see Fig. 1). Our approach builds on resampled importance sampling (RIS) [Talbot 2005], a technique for taking a set of samples that are from one distribution and selecting a weighted subset of them using another distribution that better matches the function being integrated. Unlike prior applications of RIS, we use a small fixed-size data structure—a "reservoir" that only stores accepted samples—and an associated sampling algorithm (used frequently in non-graphics applications [Efraimidis and Spirakis 2006]) to help achieve stable, real-time performance.

Given the reservoir, our approach does not use any data structures more complicated than fixed-size arrays, yet it stochastically, progressively, and hierarchically improves each pixel's direct light sampling PDF by *reusing* statistics from temporal and spatial neighbors. In contrast to modern real-time denoising algorithms that reuse *pixel colors* across temporal and spatial neighborhoods, our reuse informs the *sampling probabilities* used within the renderer, which in turn makes an unbiased algorithm possible. Our unbiased mode can be modified to be biased, which further reduces noise at the cost of some over-darkening near geometric discontinuities. We demonstrate our algorithms running interactively on a single GPU with scenes that have thousands to millions of dynamic lights, obtaining one to two orders of magnitude speedup for the same error compared to state-of-the-art methods implemented on the same hardware.

We cover the mathematical preliminaries of the techniques we build upon in Section 2 before describing our work in the subsequent sections. We discuss related work in Section 7, for better context when comparing with our results.

2 PRELIMINARIES

The reflected radiance L of a point y in direction $\vec{\omega}$ due to direct lighting is given by an integral over all light emitting surfaces A:

$$L(y,\omega) = \int_A \rho(y, \overrightarrow{yx} \leftrightarrow \vec{\omega})\, L_e(x \rightarrow y)\, G(x \leftrightarrow y)\, V(x \leftrightarrow y)\, \mathrm{d}A_x, \quad (1)$$

for BSDF ρ, emitted radiance L_e, mutual visibility V between x and y, and a geometry term G containing inverse squared distance and cosine terms. By dropping the viewing direction $\vec{\omega}$ and shading point

y for brevity and denoting differential area as $\mathrm{d}x$, this simplifies to

$$L = \int_A f(x)\,\mathrm{d}x, \quad \text{where} \quad f(x) \equiv \rho(x)\,L_e(x)\,G(x)\,V(x). \tag{2}$$

Importance Sampling (IS). Standard Monte Carlo importance sampling (IS) estimates an integral by choosing N samples x_i from a source PDF $p(x_i)$ and computing:

$$\langle L\rangle^N_{\text{is}} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \approx L. \tag{3}$$

IS remains unbiased if $p(x)$ is positive whenever $f(x)$ is non-zero, and ideally $p(x)$ is correlated with $f(x)$ to reduce variance.

Multiple Importance Sampling (MIS). In practice, directly sampling proportional to $f(x)$ is infeasible, in part due to the visibility factor $V(x)$. However, we can often draw samples proportional to individual terms in the integrand (e.g., the BSDF ρ or the emissive surfaces L_e). Given M such candidate sampling strategies p_s, MIS [Veach and Guibas 1995b] draws N_s samples from each strategy s and combines them into a single weighted estimator:

$$\langle L\rangle^{M,N}_{\text{mis}} = \sum_{s=1}^{M}\frac{1}{N_s}\sum_{i=1}^{N_s} w_s(x_i)\frac{f(x_i)}{p_s(x_i)}. \tag{4}$$

As long as the weights w_s form a partition of unity $\sum_{s=1}^{M} w_s(x) = 1$, MIS remains unbiased. The balance heuristic, $w_s(x) = \frac{N_s p_s(x)}{\sum_j N_j p_j(x)}$, is a popular and provably good choice [Veach and Guibas 1995b] for non-negative weights [Kondapaneni et al. 2019], and is equivalent to sampling from the mixture distribution of the M individual strategies.

2.1 Resampled Importance Sampling (RIS)

An alternative to sampling from a linear combination of shading terms using MIS is to sample *approximately* proportional to the product of some of the terms. Resampled importance sampling [Talbot et al. 2005] achieves this by generating $M \geq 1$ candidate samples $\mathbf{x} = \{x_1, \ldots, x_M\}$ from a source distribution p that is sub-optimal, but easy to sample from (e.g., $p \propto L_e$). It then randomly chooses an index $z \in \{1, \ldots, M\}$ from this pool of candidates with discrete probabilities

$$p(z \mid \mathbf{x}) = \frac{\mathrm{w}(x_z)}{\sum_{i=1}^{M}\mathrm{w}(x_i)} \quad \text{with} \quad \mathrm{w}(x) = \frac{\hat{p}(x)}{p(x)}, \tag{5}$$

driven by a desired target PDF $\hat{p}(x)$, for which no practical sampling algorithm may exist (e.g., $\hat{p} \propto \rho \cdot L_e \cdot G$). (Note we use 'w' for the RIS weights, to distinguish from MIS weights 'w'.) A sample $y \equiv x_z$ is selected and used in the 1-sample RIS estimator:

$$\langle L\rangle^{1,M}_{\text{ris}} = \frac{f(y)}{\hat{p}(y)}\cdot\left(\frac{1}{M}\sum_{j=1}^{M}\mathrm{w}(x_j)\right). \tag{6}$$

Intuitively, the estimator uses y as if it were drawn from $\hat{p}$ and then uses the parenthesized factor to correct for the fact that the true distribution of y only approximates $\hat{p}$.

Repeating RIS multiple times and averaging the results yields an N-sample RIS estimator:

$$\langle L\rangle^{N,M}_{\text{ris}} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(y_i)}{\hat{p}(y_i)}\cdot\left(\frac{1}{M}\sum_{j=1}^{M}\mathrm{w}(x_{ij})\right)\right). \tag{7}$$

RIS is unbiased as long as $M, N \geq 1$ and the functions p and $\hat{p}$ are positive wherever f is non-zero. While M and N can be chosen freely, there exists an optimal ratio of M to N determined by the variance and relative cost of $\hat{p}$ and f [Talbot et al. 2005]. In practice, determining this ratio a-priori can be challenging, and the optimal number of candidate samples M per sample y_i may be determined empirically instead. From now on, we will assume $N = 1$ for simplicity; our estimators can be trivially extended to the $N > 1$ case by averaging N independent executions, each with M independent candidate samples.

Generally, each pixel q in the image will have its own unique integrand f_q and corresponding target PDF $\hat{p}_q$; we denote this dependence with a subscript from here on. We show pseudo-code for RIS in Alg. 1.

Algorithm 1: Resampled importance sampling.

Input : M, q: number of candidates to generate ($M \geq 1$) for pixel q.
Output: Sample y and the sum of RIS weights $\sum_{i=1}^{M}\mathrm{w}(x_i)$

```
// Generate proposals x = {x_1, ..., x_M}
x ← ∅
w ← ∅
w_sum ← 0
for i ← 1 to M do
    generate x_i ~ p
    x ← x ∪ {x_i}
    w_i ← p̂_q(x_i)/p(x_i)
    w_sum ← w_sum + w_i
    w ← w ∪ {w_i}
// Select from candidates x
Compute normalized CDF C from w
draw random index z ∈ [0, M) using C to sample ∝ w_z
y ← x_z
return y, w_sum
```

Combining RIS with MIS. Above we assumed a single source PDF p, but many problems have several reasonable sampling techniques (e.g., BSDF or light sampling). As long as p is positive anywhere $\hat{p}$ is positive, the distribution of y approaches $\hat{p}$ as $M \to \infty$ [Talbot 2005]. However, the shape of the source PDF p influences both the effective PDF of y and the speed it converges to $\hat{p}$. In practice, when a target PDF $\hat{p}$ is the product of two functions (e.g., lighting × BSDF), the effective PDF of y will vary depending on which function proposals are drawn from (lighting or BSDF).

Luckily, Talbot [2005] showed how to leverage multiple competing techniques using MIS within RIS to reduce variance: generate the pool of proposals using MIS and use the effective MIS (mixture) PDF as the source PDF in the rest of the RIS procedure.

Unfortunately, the cost of this form of MIS increases quadratically with the number of techniques (since weights need to be

Algorithm 2: Weighted reservoir sampling.

```
class Reservoir
    y ← 0 // The output sample
    w_sum ← 0 // The sum of weights
    M ← 0 // The number of samples seen so far
    function update(x_i, w_i)
        w_sum ← w_sum + w_i
        M ← M + 1
        if rand() < (w_i / w_sum) then
            y ← x_i
function reservoirSampling(𝕊)
    Reservoir r
    for i ← 1 to M do
        r.update(𝕊[i], weight(𝕊[i]))
    return r
```

evaluated for each proposal and each such weight needs to consider all proposal PDFs). This is not a problem when MIS is used with just two techniques (e.g., lighting and BSDF), but it quickly becomes intractable as the number of strategies increases.

We use RIS in a way that increases the number of candidates dramatically through spatial and temporal reuse, each using different source PDFs and integration domains. We rederive RIS in this more general setting in Section 4, and introduce a new MIS approach that is computationally tractable.

2.2 Weighted Reservoir Sampling

Weighted reservoir sampling (WRS) [Chao 1982] is a family of algorithms for sampling N random elements from a stream $\{x_1, x_2, x_3, \ldots, x_M\}$ in a single pass over the data. Each element has an associated weight $\mathrm{w}(x_i)$ such that x_i should be selected with probability

$$P_i = \frac{\mathrm{w}(x_i)}{\sum_{j=1}^{M} \mathrm{w}(x_j)}. \tag{8}$$

Reservoir sampling processes each element exactly once, and only the N items in the reservoir must remain in memory. The stream length M need not be known in advance.

Reservoir sampling algorithms are classified based on whether element x_i may appear multiple times in the output set, i.e. if samples are chosen *with* or *without* replacement. Literature has mostly focused on sampling without replacement, as it is a fundamentally more difficult problem. Fortunately, we want independent selections x_i for Monte Carlo integration, so we only consider weighted reservoir sampling *with* replacement below.

Reservoir sampling processes elements of an input stream in order, storing a *reservoir* of N samples. At any point in the stream, reservoir sampling maintains the invariant that samples in the reservoir are drawn from the desired distribution (over all elements processed thus far). When the stream ends, the reservoir is returned. In the following, we focus on the case where $N = 1$, i.e. where the reservoir consists of one sample.

When processing a new stream element, the reservoir is updated so as to maintain the invariant, which is that after m samples have been processed, sample x_i occurs in the reservoir with probability

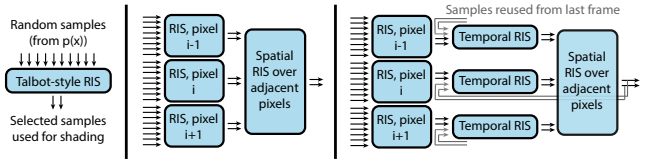


Fig. 3. (Left) Talbot et al. [2005] RIS selects a few samples from a larger pool of randomly-selected candidates. (Center) RIS can be viewed as an abstract building block selecting a subset of its inputs. Combining blocks in sequence can reuse (and amortize costs of generating) the random input candidates over multiple pixels. (Right) Samples can also be reused temporally, giving an effective sample count (M in Eq. (7)) that grows based on the spatial and temporal filter sizes.

$\mathrm{w}(x_i)/\sum_{j=1}^{m} \mathrm{w}(x_j)$. The update rule stochastically replaces x_i in the reservoir with the next sample x_{m+1}, with probability

$$\frac{\mathrm{w}(x_{m+1})}{\sum_{j=1}^{m+1} \mathrm{w}(x_j)}, \tag{9}$$

which ensures that x_{m+1} appears in the reservoir with the desired frequency. Thus, any previous sample x_i is in the reservoir with probability

$$\frac{\mathrm{w}(x_i)}{\sum_{j=1}^{m} \mathrm{w}(x_j)} \left(1 - \frac{\mathrm{w}(x_{m+1})}{\sum_{j=1}^{m+1} \mathrm{w}(x_j)}\right) = \frac{\mathrm{w}(x_i)}{\sum_{j=1}^{m+1} \mathrm{w}(x_j)}, \tag{10}$$

which also maintains the invariant.

This algorithm was introduced by Chao [1982], and is outlined in Alg. 2. It only stores the sample in the reservoir and a running sum of weights, making it very efficient.

3 STREAMING RIS WITH SPATIOTEMPORAL REUSE

RIS and WRS form the foundation of our algorithm, and together allow us to process random candidates in a streaming fashion while keeping our algorithm and data structures extremely simple (Section 3.1). Given such a streaming algorithm, we show how a property of WRS allows us to do *spatiotemporal resampling* to efficiently combine and reuse candidates from neighboring pixels and even past frames (Section 3.2). Doing so increases our effective sample count by orders of magnitude (see Fig. 3) with little added computation.

Unfortunately, the naive approach to spatiotemporal resampling is biased, as different pixels select samples based on different BRDFs and surface orientations. This leads to energy loss near geometric discontinuities in images, similar to problems typical in post-process filtering. In Section 4, we show how to generalize RIS and use an MIS reweighting of the varying sample PDFs to maintain unbiasedness.

3.1 Streaming RIS using reservoir sampling

It is straightforward to apply the WRS algorithm to RIS to transform it into a streaming algorithm, by updating the reservoir with sequentially generated candidates x_i and corresponding weights (Alg. 3). In Figure 4, we show an image from our GPU implementation of streaming RIS for direct lighting in a complex scene with 23,000 emissive triangles. We generate samples uniformly over the area of emitters and use the unshadowed path contribution $\hat{p}(x) = \rho(x)\, L_e(x)\, G(x)$

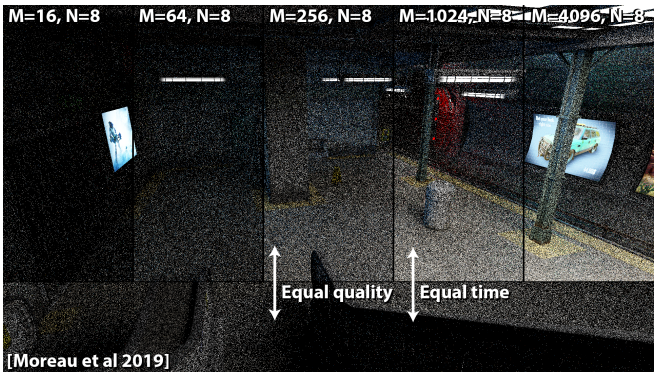


Fig. 4. Streaming RIS quality improves with increased M (candidates) and N (samples for shading). Here we show the effect of increasing M in the multi-room Subway scene with 23,000 textured emissive triangles. Tracing 8 shadow rays costs 6 ms; selecting those samples costs (left to right) 1.0, 2.5, 10.1, 42, and 168 ms. Moreau et al. [2019]'s total cost is 48 ms when shooting 8 rays, comparable to $M = 1024$, but with quality comparable to $M = 256$. Subway ©silvertm

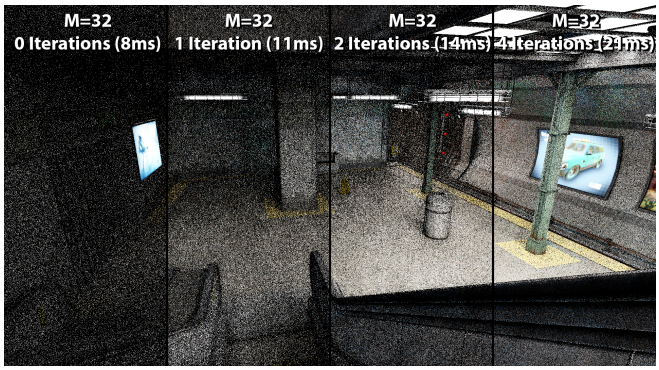


Fig. 5. Starting from $m = 32$ candidates generated by streaming RIS (left), we iteratively apply our spatial reuse operation, gathering $k = 5$ neighbors at each step. The number of repeated applications increase from left to right with 1, 2 and 4 iterations respectively. The image quality increases dramatically without much added cost. Subway ©silvertm

Algorithm 3: Streaming RIS using weighted reservoir sampling.

```
1  foreach pixel q ∈ Image do
2      Image[q] ← shadePixel(RIS(q), q)
3  function RIS(q)
4      Reservoir r
5      for i ← 1 to M do
6          generate x_i ~ p
7          r.update(x_i, p̂_q(x_i)/p(x_i))
8      r.W = 1/p̂_q(r.y) (1/r.M r.w_sum)   // Equation (6)
9      return r
10 function shadePixel(Reservoir r, q)
11     return f_q(r.y) · r.W
```

as the target distribution, only tracing shadow rays for the N surviving RIS samples. We compare streaming RIS with varying candidate counts M to a reference as well as to a state-of-the-art real-time light BVH [Moreau et al. 2019] using an equal number of rays per pixel.

Surprisingly, as M increases, streaming RIS beats even a state-of-the-art light sampling technique, without preprocessing or relying on a complex data structure. However, good results require large values of M. While Alg. 3 makes the *storage* requirements constant (from $O(M)$), *computation* remains linear in M.

3.2 Spatiotemporal Reuse

The approach described in Section 3.1 independently generates candidates at each pixel q and resamples them using a target PDF $\hat{p}_q$. A key observation is that significant correlation generally exists between target PDFs in neighboring pixels. For example, if using unshadowed illumination ($\hat{p}(x) = \rho(x)\, L_e(x)\, G(x)$), then spatial proximity often leads to the geometry and BSDF factors being similar at adjacent pixels. A naive way to leverage correlations between "similar" pixels would be to generate (and store) per-pixel candidate samples and their weights and to use a second pass to *reuse* computation performed at neighboring pixels by combining each pixel's candidates with its neighbors'. Because weight computations occur in the first pass, reuse of neighbors' candidates are computationally cheaper than generating an equivalent number of new candidates. (This is similar to Bekaert et al. [2002]'s reuse, though they retrace visibility rays for reused candidates.)

Unfortunately this approach is impractical, as it requires *storage* for each reused candidate. However, we can circumvent the storage requirements using a key property of reservoir sampling, which allows us to combine multiple reservoirs without requiring access to their input streams.

A reservoir's state contains both the currently selected sample y and the sum of weights w_{sum} of all candidates seen thus far. To combine two reservoirs, we treat each reservoir's y as a fresh sample with weight w_{sum}, and feed it as input to a new reservoir. The result is mathematically equivalent to having performed reservoir sampling on the two reservoirs' combined input streams. However, crucially this operation only requires *constant* time and avoids storing (or retrieving) elements of either input stream, needing only access to each reservoir's current state. Input streams of an arbitrary number of reservoirs can be combined this way: Alg. 4 shows pseudocode to combine the input streams of k reservoirs; it runs in $O(k)$ time. To account for the fact that samples from the neighboring pixel q' are resampled following a different target distribution $\hat{p}_{q'}$, we reweight the samples with the factor $\hat{p}_q(r.y)/\hat{p}_{q'}(r.y)$ to account for areas that were over- or undersampled at the neighbor compared to the current pixel. The resulting term $\hat{p}_q(r.y)/\hat{p}_{q'}(r.y) \cdot r.w_{sum}$ can be written more succinctly as $\hat{p}_q(r.y) \cdot r.W \cdot r.M$ using the term already computed in Alg. 3, line 8.

Spatial Reuse. This property of reservoir sampling makes possible a practical algorithm for reusing computation in RIS. We first generate M candidates for every pixel q using RIS(q) (Alg. 3) and store the resulting reservoirs in an image-sized buffer. In a second step, each pixel selects k of its neighbors and combines their reservoirs with its

Algorithm 4: Combining the streams of multiple reservoirs.

Input : Reservoirs r_i to combine.
Output: A combined reservoir equivalent to the concatenated input streams of $r_1, \ldots, r_k$.

function combineReservoirs($q, r_1, r_2, \ldots, r_k$)
  Reservoir s
  foreach $r \in \{r_1, \ldots, r_k\}$ **do**
    s.update($r.y$, $\hat{p}_q(r.y) \cdot r.W \cdot r.M$)
  $s.M \leftarrow r_1.M + r_2.M + \ldots + r_k.M$
  $s.W = \frac{1}{\hat{p}_q(s.y)} \left(\frac{1}{s.M} s.\mathrm{w}_{\mathrm{sum}}\right)$ // Equation (6)
  return s

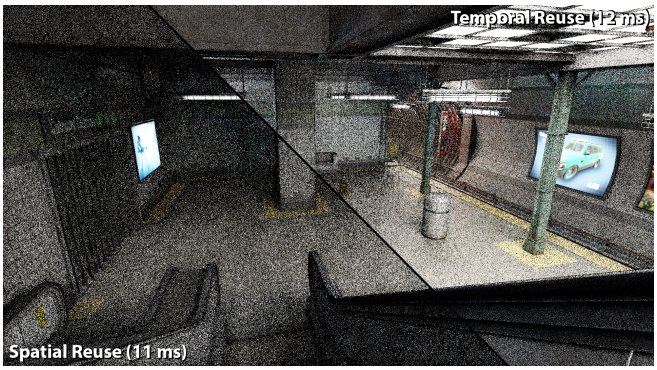


Fig. 6. Compared to one iteration of spatial reuse alone (left, $M = 4$, $k = 5$), adding candidates from previous frames to candidates from the current frame can greatly increase the image quality of streaming RIS (right, after 20 frames) with little added computational cost. Subway ©silvertm

own using Alg. 4. Per pixel costs are $O(k + M)$, but each pixel *effectively* sees $k \cdot M$ candidates. Crucially, spatial reuse can be *repeated*, using the outputs of the prior reuse pass as input. Performing n iterations requires $O(nk + M)$ computation, but effectively yields $k^n M$ candidates per pixel, assuming distinct neighboring pixels are used at each step.

Figure 5 shows spatial reuse in the Subway scene. Each iteration requires little additional computation, but dramatically increases image quality. The benefit is not indefinite; eventually, iterative reuse incorporates all candidates from nearby pixels and image quality stops improving.

Temporal Reuse. Images are often not rendered in isolation but are part of an animated sequence. In this case, the prior frame can provide additional candidates for reuse. After rendering a frame, we store each pixel's final reservoir for reuse in the next frame. If we render frames sequentially and feed forward their reservoirs, a frame combines candidates not just with those of the previous frame, but *all* previous frames in the sequence, which dramatically improves image quality. Figure 6 again shows the Subway scene, comparing spatial-only and spatiotemporal reuse.

Visibility Reuse. Unfortunately, even with an unlimited number of candidates, RIS cannot achieve noise-free renderings. Although the distribution of samples approaches the target PDF $\hat{p}$ as M approaches

Algorithm 5: Our algorithm for RIS with spatiotemporal reuse.

Input : Image sized buffer containing the previous frame's reservoirs
Output: The current frame's reservoirs

function reservoirReuse(prevFrameReservoirs)
  reservoirs ← new Array[ImageSize]
  // Generate initial candidates
  foreach *pixel* $q \in$ Image **do**
    reservoirs[q] ← RIS(q) // Alg. 3
  // Evaluate visibility for initial candidates
  foreach *pixel* $q \in$ Image **do**
    if shadowed(reservoirs[q].y) **then**
      reservoirs[q].$W \leftarrow 0$
  // Temporal reuse
  foreach *pixel* $q \in$ Image **do**
    $q' \leftarrow$ pickTemporalNeighbor(q)
    reservoirs[q] ← combineReservoirs(q, reservoirs[q], prevFrameReservoirs[q']) // Alg. 4
  // Spatial reuse
  for *iteration* $i \leftarrow 1$ *to* n **do**
    foreach *pixel* $q \in$ Image **do**
      $Q \leftarrow$ pickSpatialNeighbors(q)
      $\mathbb{R} \leftarrow \{\text{reservoirs}[q'] \mid q' \in Q\}$
      reservoirs[q] ← combineReservoirs(q, reservoirs[q], $\mathbb{R}$)
  // Compute pixel color
  foreach *pixel* $q \in$ Image **do**
    Image[q] ← shadePixel(reservoirs[q], q) // Alg. 3
  return reservoirs

infinity, $\hat{p}$ does not sample the integrand f perfectly. In practice, $\hat{p}$ is usually set to the unshadowed path contribution, meaning that as M grows large, noise due to visibility starts to dominate. Unfortunately, visibility noise can be severe in large scenes. To solve this issue, we also perform *visibility reuse*. Before performing spatial or temporal reuse, we evaluate visibility of the selected sample y for each pixel's reservoir. If y is occluded, we discard the reservoir. This means that occluded samples will not propagate to neighboring pixels, and if visibility is locally coherent, the final sample produced by spatial resampling is likely to be unoccluded.

Alg. 5 provides pseudocode for our complete algorithm. We first generate and resample from M independent per-pixel light candidates. The selected samples from this step are tested for visibility, and occluded samples discarded. We then combine the selected samples in each pixel's reservoir with the prior frame's output, determined using backprojection. We perform n rounds of spatial reuse to leverage information from a pixel's neighbors. Finally, we shade the image and forward the final reservoirs to the next frame.

4 (ELIMINATING) BIAS IN MULTI-DISTRIBUTION RIS

In the previous section, we introduced a practical algorithm to reuse computation, spatially and temporally, that dramatically improves the quality of RIS with low overhead. However, we ignored one important detail: Each pixel uses a different integration domain and target distribution, and reusing candidates from adjacent pixels can

potentially introduce bias. This is because the PDF of samples after resampling varies from pixel to pixel due to the different target distributions. Standard RIS is not designed to accomodate mixing candidate samples from different PDFs as we do during reuse, and ignoring this fact can lead to noise and bias.

The rest of this section is structured as follows: In Section 4.1–Section 4.3, we rederive and do a theoretical analysis of RIS in the presence of candidates generated from different PDFs, and reveal the source of this bias as well as a simple solution to retain unbiasedness. Readers less interested in theory can skip directly to Section 4.4, in which we detail the practical changes to our algorithm needed to accomodate our theory.

4.1 Analyzing the RIS Weight

To illustrate the source of bias in RIS, we begin by regrouping Eq. (6) as follows:

$$\langle L \rangle_{\text{ris}}^{1,M} = f(y) \cdot \left(\frac{1}{\hat{p}(y)} \frac{1}{M} \sum_{j=1}^{M} \text{w}(x_j) \right) = f(y) W(\mathbf{x}, z), \tag{11}$$

where W is the stochastic weight for the generated sample $y \equiv x_z$:

$$W(\mathbf{x}, z) = \frac{1}{\hat{p}(x_z)} \left[\frac{1}{M} \sum_{i=1}^{M} \text{w}_i(x_i) \right]. \tag{12}$$

What is the role of W? Normally, Monte Carlo estimators take on the form $f(y)/p(y)$. We do not know $p(y)$—in fact, we later show that we cannot compute it in closed form—and $W(\mathbf{x}, z)$ takes its place in Eq. (11). We can therefore guess that $W(\mathbf{x}, z)$ must take on the role of the reciprocal PDF $1/p(y)$. However, $W(\mathbf{x}, z)$ is a random variable: For a given output sample y there are many $\{\mathbf{x}, z\}$ that could have produced it, and which set of values (and therefore, which value for $W(\mathbf{x}, z)$) is returned by RIS is random.

In order for Eq. (6) to be unbiased, the expected value of $W(\mathbf{x}, z)$ should be equal to $1/p(y)$. In the following sections, we show that this is not always the case when combining samples from neighboring pixels, which is the source of bias.

Explanation of Reweighting Factor. In Alg. 4, samples from neighbors are assigned the weight $\hat{p}_q(r.y) \cdot r.W \cdot r.M$. We gave an intuitive justification of this weight in Section 3.2, but this term now has a straightforward explanation: $\hat{p}_q(r.y) \cdot r.W$ simply represents the standard RIS weight of $\hat{p}_q(r.y)/p(r.y)$, except that we do not know the exact PDF $p(r.y)$ and use the estimate of the inverse PDF, $r.W$ (Eq. (12)), instead. As $r.y$ represents the result of combining multiple samples, the weight is additionally scaled by the number of candidates $r.M$ that produced $r.y$.

4.2 Biased RIS

We will now derive the effective PDF $p(y)$ of samples produced by RIS. Standard RIS [Talbot et al. 2005] (Section 2.1) assumes that all candidate samples are produced by the same pdf p. We instead now allow each sample x_i in $\mathbf{x}$ to come from a potentially different source PDF $p_i(x_i)$. The joint PDF of these proposals is simply the product of their PDFs:

$$p(\mathbf{x}) = \left[\prod_{i=1}^{M} p_i(x_i) \right]. \tag{13}$$

In the second stage of the RIS algorithm, we pick a discrete index $z \in \{1, \dots, M\}$, but with selection probabilities and weights now driven by these candidate-specific PDFs (cf. Eq. (5)):

$$p(z \mid \mathbf{x}) = \frac{\text{w}_z(x_z)}{\sum_{i=1}^{M} \text{w}_i(x_i)} \qquad \text{where} \qquad \text{w}_i(x) = \frac{\hat{p}(x)}{p_i(x)}. \tag{14}$$

Since we have $p(\mathbf{x})$ and $p(z \mid \mathbf{x})$, we can easily write down the joint PDF of the candidates $\mathbf{x}$ and selected index z as the product:

$$p(\mathbf{x}, z) = p(\mathbf{x})\, p(z \mid \mathbf{x}) = \left[\prod_{i=1}^{M} p_i(x_i) \right] \frac{\text{w}_z(x_z)}{\sum_{i=1}^{M} \text{w}_i(x_i)}. \tag{15}$$

So what is $p(y)$? For a fixed output sample y, there are potentially many configurations of $\mathbf{x}$ and z that could lead to y being returned by RIS. For example, we could have $x_1 = y$ and $z = 1$ and all other $x_2, \dots, x_M$ chosen freely. We could also have $x_2 = y$ and $z = 2$, and so forth. Of course, y can only be produced by techniques for which $p_i(y) > 0$. Let's gather these techniques into a set

$$Z(y) = \{ i \mid 1 \le i \le M \wedge p_i(y) > 0 \}. \tag{16}$$

To obtain the total PDF of an output sample y, we simply marginalize the joint PDF (15) over all configurations that could lead to this y:

$$p(y) = \sum_{i \in Z(y)} \underbrace{\int \cdots \int}_{M-1 \text{ times}} p(\mathbf{x}^{i \to y}, i) \underbrace{\mathrm{d}x_1 \dots \mathrm{d}x_M}_{M-1 \text{ times}}. \tag{17}$$

where $\mathbf{x}^{i \to y} = \{x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_M\}$ is shorthand for the set of candidates with the ith candidate fixed to y. The integration is only over the $M - 1$ candidates that are not fixed.

Expected RIS Weight. With the PDF of RIS defined, we can now show when the expected value of the RIS weight $W(\mathbf{x}, z)$ is the PDF's reciprocal. To compute this value, we need to take a conditional expectation: *Given* that the output sample is y, what is the average weight? We can do this by taking the expectation of $W(\mathbf{x}, z)$ only over those values of $\mathbf{x}$ and z for which $x_z = y$, and divide by $p(y)$: the probability density of the event $x_z = y$. This gives

$$\mathop{\mathbb{E}}_{x_z = y}[W(\mathbf{x}, z)] = \sum_{i \in Z(y)} \frac{\int \cdots \int W(\mathbf{x}^{i \to y}, i)\, p(\mathbf{x}^{i \to y}, i)\, \mathrm{d}x_1 \dots \mathrm{d}x_M}{p(y)}, \tag{18}$$

where $\mathbf{x}^{i \to y}$ and the integration bounds are the same as in Eq. (17).

In Appendix A we prove that this expression simplifies to:

$$\mathop{\mathbb{E}}_{x_z = y}[W(\mathbf{x}, z)] = \frac{1}{p(y)} \frac{|Z(y)|}{M}, \tag{19}$$

which shows two things: If all candidate PDFs are non-zero wherever the target function is non-zero, then $|Z(y)| = M$, and the RIS weight becomes an unbiased estimator of the inverse RIS PDF. If, however, some of the PDFs are zero for part of the integrand, then $\frac{|Z(y)|}{M} < 1$, and the inverse PDF is consistently underestimated. This means the expected value is biased to be darker than the true integral.

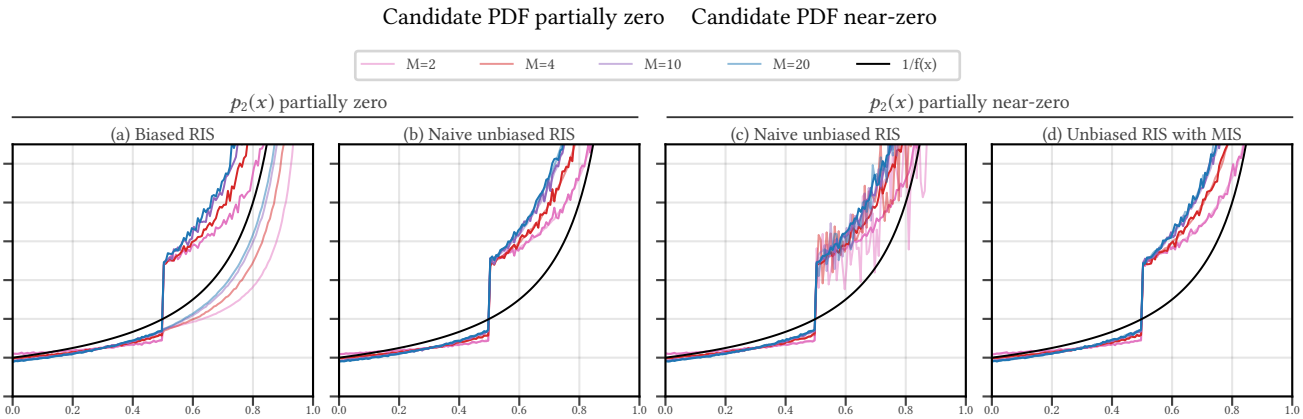
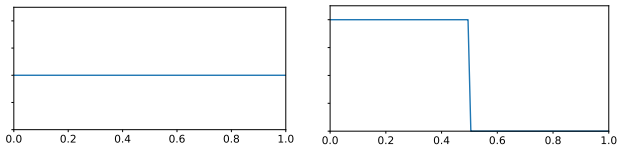


Fig. 7. We show results of RIS for sampling a simple linear target PDF, $\hat{p}(x) = 2 - 2x$. Candidates are produced from a constant PDF ($p_1(x) = 1$) and a step function ($p_2(x) = 2H(1/2 - x)$). We show the inverse PDF of samples produced by RIS, both estimated from the histogram of output samples (dark, thick lines; this is the ground truth), and estimated by the RIS weight (pale lines). The traditional RIS weight (a) is biased where one or more of the PDFs are zero (right half of graph), and the RIS weight (pale lines) does not match the actual distribution of samples (dark lines). Naive unbiased RIS (b) fixes the bias by dividing by the number of non-zero candidate PDFs rather than M, but this strategy leads to an extremely noisy RIS weight (c) when a candidate PDF is near-zero rather than zero ($p_2(x) \propto \max(2H(1/2 - x), 10^{-4})$). Our MISed version of the RIS weight (d) is unbiased and robust against small candidate PDFs.

A 1D Example. To demonstrate this, consider the following two candidate PDFs: $p_1(x) = 1$ and $p_2(x) = 2H(1/2 - x)$, where $H(x)$ is the Heaviside step function. The candidate PDFs are illustrated below:



In Fig. 7(a), we used these two candidate PDFs to sample a linear ramp, $\hat{p}(x) = 2 - 2x$, with half the candidates generated from p_1 and the others from p_2, for increasing values of M. We visualized $1/p(y)$, measured in two different ways: once, by plotting the *reciprocal of the histogram* of sample locations (solid, dark curves; this is the ground truth), and once as the average of the RIS weight at each location (pale, transparent curves). The curves do not match, but if standard RIS were truly an estimator of the inverse PDF they should.

4.3 Unbiased RIS

We now show that this bias can be eliminated by modifying the RIS weight: Instead of multiplying by the factor $1/M$, we can choose some (yet unspecified) weight $m(x_z)$:

$$W(\mathbf{x}, z) = \frac{1}{\hat{p}(x_z)} \left[m(x_z) \sum_{i=1}^{M} \mathrm{w}_i(x_i) \right]. \tag{20}$$

Repeating the derivation of the expected value of W shows that

$$\mathop{\mathbb{E}}_{x_z=y}[W(\mathbf{x}, z)] = \frac{1}{p(y)} \sum_{i \in Z(y)} m(x_i), \tag{21}$$

indicating an unbiased estimator just requires $\sum_{i \in Z(y)} m(x_i) = 1$.

Naive approach. There are infinitely many ways to choose $m(x)$. The easiest way is to use uniform weights and simply set $m(x_z) = 1/|Z(x_z)|$. That is, instead of dividing by M (the number of candidates), we divide by the number of candidates with non-zero PDFs at that location, creating an unbiased RIS estimator (see Fig. 7(b)).

This fixes the bias problem; but, this estimator of the inverse PDF can have problems. Consider a candidate PDF close to, but not exactly, zero such as $p_2(x) \propto \max(H(1/2 - x), 10^{-4})$. As the candidate PDF is never zero, even the original RIS estimator will be unbiased. However, the estimator of the inverse RIS PDF becomes extremely noisy, as shown in Fig. 7(c).

Combining with Multiple Importance Sampling. Luckily, we are able to choose any weights $m(x_z)$ that sum to 1, for instance:

$$m(x_z) = \frac{p_z(x_z)}{\sum_{i=1}^{M} p_i(x_z)}, \tag{22}$$

i.e., the balance heuristic of the candidate PDFs. This solves both bias and noise issues when combining many candidate PDFs using RIS, as shown in Fig. 7(d).

Comparison to Talbot et al. [2005]. Talbot et al. propose a different solution for using multiple candidate PDFs in RIS. Where we use $w_i(x) = \hat{p}(x)/p_i(x)$ (Eq. (14)) as the weight, Talbot et al. use $w_i(x) = \hat{p}(x)/\sum p_i(x)$. By replacing the individual PDFs by a single average PDF, Talbot forgo noise and bias issues that arise when mixing multiple candidate PDFs. In addition, if the sum of candidate PDFs is closer to the target distribution than the individual PDFs, then Talbot et al.'s approach may further reduce noise compared to ours. However, there is a crucial difference between the two approaches: Talbot et al. evaluate *all* PDFs for *each* candidate sample; if each candidate sample uses a different PDF, then the cost of their approach is $O(M^2)$ PDF evaluations. In contrast, our approach evaluates only one PDF for each candidate, and all PDFs only once more when computing the final MIS weight (Eq. (22)), equivalent to a cost of $O(M)$. This is especially crucial in our case, in which evaluating

Algorithm 6: Unbiased combination of multiple reservoirs.

Input : Reservoirs r_i and the pixels q_i they originated from.
Output: An unbiased combination of the input reservoirs.

function combineReservoirsUnbiased($q, r_1, r_2, \ldots, r_k, q_1, \ldots, q_k$)
  Reservoir s
  foreach $r \in \{r_1, \ldots, r_k\}$ **do**
    s.update($r.y$, $\hat{p}_q(r.y) \cdot r.W \cdot r.M$)
  $s.M \leftarrow r_1.M + r_2.M + \ldots + r_k.M$
  $Z \leftarrow 0$
  foreach $q_i \in \{q_1, \ldots, q_k\}$ **do**
    if $\hat{p}_{q_i}(s.y) > 0$ **then**
      $Z \leftarrow Z + r_i.M$
  $m \leftarrow 1/Z$
  $s.W = \frac{1}{\hat{p}_q(s.y)} (m \cdot s.\mathrm{w}_{\mathrm{sum}})$ // Equation (20)
  return s

the PDF may involve *tracing a ray*; the quadratic cost of Talbot et al.'s approach then makes it completely infeasible in this use case, whereas the linear cost of our approach offers unbiasedness at affordable cost. In the supplemental material, we offer more detailed discussion and empirical comparison between the two approaches to further demonstrate this point.

4.4 A Practical Algorithm for Unbiased Reuse

We can now apply our bias correction to our algorithm for sample reuse (Alg. 5). The bias is introduced when combining multiple reservoirs (Alg. 4): a pixel q gathers reservoirs r_i from its neighboring pixels, each of which contributes a sample $r_i.y$; however, the PDF of this sample may be zero where the integrand at q is not. For example, candidates that lie below the hemisphere are normally discarded. However, neighboring pixels may have differently oriented surface normals, and may discard samples that would have non-zero contribution at q. Similarly, our algorithm discards samples that are occluded after the first round of resampling (effectively setting the PDF to zero); however, a sample occluded at one pixel may be visible at its neighbor, and discarding it causes bias.

Each sample $r_i.y$ is the result of resampling, and we do not know its true PDF (since Equation (17) cannot be evaluated in closed form). However, as long as we know an approximate form of this PDF that is zero whenever the real PDF is zero, we can use it instead to compute an unbiased weight. For pixel q_i, we use $\hat{p}_{q_i}(x)$ as an approximation to the real PDF of samples at q_i, as it is zero wherever the true PDF is. If visibility reuse is employed, we additionally check if x is occluded at q_i, and set the PDF to zero if it is (as such samples are discarded).

We give pseudocode for our unbiased reservoir combination (with uniform weights) in Alg. 6; the MIS version is analogous. Unfortunately, the unbiased version can be significantly more expensive: if we employ visibility reuse, then $\hat{p}_{q_i}$ includes visibility, and evaluating it requires tracing an additional shadow ray. E.g. in spatial reuse, this means tracing k additional rays (one per neighboring pixel).

Because of this, we implemented both biased and unbiased forms of our algorithm. The biased algorithm introduces darkening whenever neighbors (temporally or spatially) have different occlusion or surface orientation. This bias can be partially avoided by choosing neighbors carefully, which we describe in the next section. Where the remaining bias is still unacceptable, our unbiased algorithm may be used, at the cost of tracing additional rays.

5 DESIGN AND IMPLEMENTATION CHOICES

We implemented both biased and unbiased variants of our algorithm in a GPU-based real-time rendering system. We have made various design choices to improve robustness and performance, as well as to limit the impact of bias, which we detail in this section. We also specify the parameters used in our implementation. In general our unbiased algorithm is computationally more expensive, and we choose different parameters for our biased and unbiased variants such that they have approximately equal cost.

Candidate Generation. We sample $M = 32$ initial candidates by importance sampling emissive triangles based on their power, and then uniformly generate a point x on the selected triangle (i.e. $p(x) \propto L_e(x)$). If an environment map is present in the scene, 25% of candidates are instead generated by importance sampling the environment map. Importance sampling for both triangles and environment map locations is accelerated using an alias table [Walker 1974]. We also experimented with pregenerating a list of VPLs on emissive triangles. Doing so yields higher performance at the cost of some visual artifacts, and may be an option for real-time applications with limited render-times. It would also be possible to use higher quality samples as initial candidates—such as those produced by the data structure of Moreau et al. [2019]—but this proved to significantly increase runtime in our preliminary tests.

Target PDF. At each resampling step in our algorithm, we weight samples based on a target PDF. We use the unshadowed path contribution $\hat{p} \propto \rho \cdot L_e \cdot G$ as the target PDF at each pixel. We use a unified material model for all geometry in the scene, consisting of a dielectric GGX microfacet layer atop a diffuse Lambertian substrate. If more sophisticated material models are used and evaluating the BRDF for each candidate is too expensive, approximations to the BRDF may be used.

Neighbor selection. For spatial reuse, we found that deterministically selected neighbors (e.g. in a small box around the current pixel) lead to distracting artifacts, and we instead sample $k = 5$ ($k = 3$ for our unbiased algorithm) random points in a 30-pixel radius around the current pixel, sampled from a low-discrepancy sequence. As an alternative, using a hierarchical *À-Trous* sampling scheme [Dammertz et al. 2010; Schied et al. 2017] also produced promising results, at the cost of some artifacts, and may be interesting for future work. For temporal reuse, we compute motion vectors to project the current pixel's position into the previous frame, and use the pixel there for temporal reuse.

For our biased algorithm, reusing candidates from neighboring pixels with substantially different geometry/material leads to increased bias, and we use a simple heuristic to reject such pixels: we compare the difference in camera distance, and the angle between normals of the current pixel to the neighboring pixel, and reject the neighbor if either exceed some threshold (10% of current pixels depth and 25°, respectively). This strategy is similar to those used in

selective blurs for real-time denoising, and we found it to substantially reduce bias. We use $n = 2$ ($n = 1$ for our unbiased algorithm) spatial reuse passes.

Evaluated Sample Count. Our Alg. 5 assumes $N = 1$, i.e. a single sample is evaluated at the end of the frame. For higher sample counts, the algorithm can simply be repeated and the results averaged. For our unbiased algorithm, we use $N = 1$ for interactive frame-rates; our biased algorithm uses $N = 4$ instead, i.e. we store four reservoirs at each pixel. For non-interactive render times, we simply average images of independent executions of our algorithm.

Reservoir storage and temporal weighting. At each pixel, we only store the information of the pixel's reservoir: The selected sample y, the number of candidates M that contributed to the pixel, and the probabilistic weight W. For $N > 1$, we store multiple samples y and weights W at each pixel to accomodate multiple reservoirs. With temporal reuse, the number of candidates M contributing to the pixel can in theory grow unbounded, as each frame always combines its reservoir with the previous frame's. This causes (potentially stale) temporal samples to be weighted disproportionately high during resampling. To fix this, we simply clamp the previous frame's M to at most 20× of the current frame's reservoir's M, which both stops unbounded growth of M and bounds the influence of temporal information.

6 RESULTS

We protoyped our method in the open-source Falcor rendering framework [Benty et al. 2019] in order to be able to apply hardware-accelerated ray tracing. We call our algorithm *Reservoir-based Spatio-Temporal Importance Resampling*, or ReSTIR for short. We tested our technique on various scenes containing thousands to millions of emissive triangles. Renderings and timings were obtained on a GeForce RTX 2080 Ti GPU, except for the Amusement Park scene, which required use of a Titan RTX due to high memory requirements.

The render times that we report include the cost of sample generation, ray tracing and shading. We do not include G-buffer rasterization cost, as this is shared between all rendering methods (and averages 1-2 ms). We report image errors of each method compared to an unbiased reference rendered at high sample count. Errors are reported as *Relative Mean Absolute Error* (RMAE), which we found less sensitive to isolated outliers than mean squared error (MSE).

For methods using temporal reuse, our figures show the final frame in a 20 frame animation involving fast camera movement. This avoids the lower quality expected during any warm up period without providing any artificial advantage by temporally supersampling a single view. Each frame in the sequence uses the same computation budget as the final frame.

Figure 1 and Figure 9 show equal-time comparisons of our biased and unbiased spatiotemporal reuse versus a state-of-the-art real-time light sampling technique [Moreau et al. 2019]. Our technique has substantially lower error than Moreau et al.'s BVH-based approach. We found that the light BVH generally under-performs even our streaming RIS algorithm (without reuse); in all further results we use streaming RIS as the baseline for comparisons.

Our supplementary video shows real-time captures of the animated Amusement Park, Subway, Bistro, and Zero Day scenes with equal-time comparisons between various combinations of uniform sampling, Moreau et al. [2019]'s approach, our biased and unbiased methods, and offline-rendered reference animations.

Figure 8 compares the biased and unbiased versions of our spatiotemporal reuse with RIS [Talbot et al. 2005] at equal time. To allow for a fair baseline comparison, we compare against our streaming version of RIS, as we found it consistently faster (20%-30% speedup) than non-streaming implementations. Our methods employing spatial and temporal reuse significantly outperform RIS without reuse, both visually and in terms of error. In some scenes (e.g. Subway), the baseline image is barely recognizable, but our spatiotemporal reuse image is nearly converged. In all scenes, our biased method has considerably less variance, at the cost of some energy loss and image darkening. The energy loss is most pronounced in regions with difficult lighting, e.g. shadow boundaries, sharp highlights and complex geometry such as trees.

Figure 11 shows how the RMAE evolves with increased render time for six different methods: sampling lights according to power and then applying MIS [Veach and Guibas 1995b] with BRDF and area-weighted sampling; Moreau et al. [2019]'s light BVH; streaming RIS, as well as three versions of our algorithms: biased and unbiased spatiotemporal reuse, as well as biased spatial reuse without temporal reuse. The last variant makes it possible to evaluate our algorithm for still images. In all scenes, our biased spatiotemporal reuse has the lowest error at *interactive* render times, usually by a significant margin. However, as render time increases, the error due to bias dominates, so our unbiased spatiotemporal reuse eventually exhibits lower error (usually at around 1 s). In most scenes, biased spatial reuse also offers competitive performance without relying on knowledge from prior frames. The lack of temporal history also limits bias propagation, and at longer render times this method can overtake biased spatiotemporal reuse due to reduced bias. In all scenes, we significantly outperform prior work.

To demonstrate the performance of our method at non-interactive render times, we compare streaming RIS and our methods on the Amusement Park scene at 1 s render time in Figure 10. Even at comparatively high render times, we still significantly outperform the baseline. Our biased spatiotemporal reuse is nearly noise-free, but the bias is apparent; if problematic, unbiased spatiotemporal reuse offers similar performance with slightly higher variance.

7 RELATED WORK

A wide range of prior approaches have addressed light sampling and sample reuse in rendering or have developed mathematical tools related to our work.

Many-light sampling. Direct lighting alone can be challenging, especially in scenes with large collections of complex emitters. Ward [1994] and Shirley et al. [1996] pioneered this area, classifying lights as 'important' and 'unimportant' based on their expected contributions. Renderers targeting scenes with many emitters today extend this idea by using light hierarchies [Estevez and Kulla 2018; Yuksel 2019] to importance sample from many lights in sub-linear time. Recent work demonstrates hierarchies can be effective for real-time

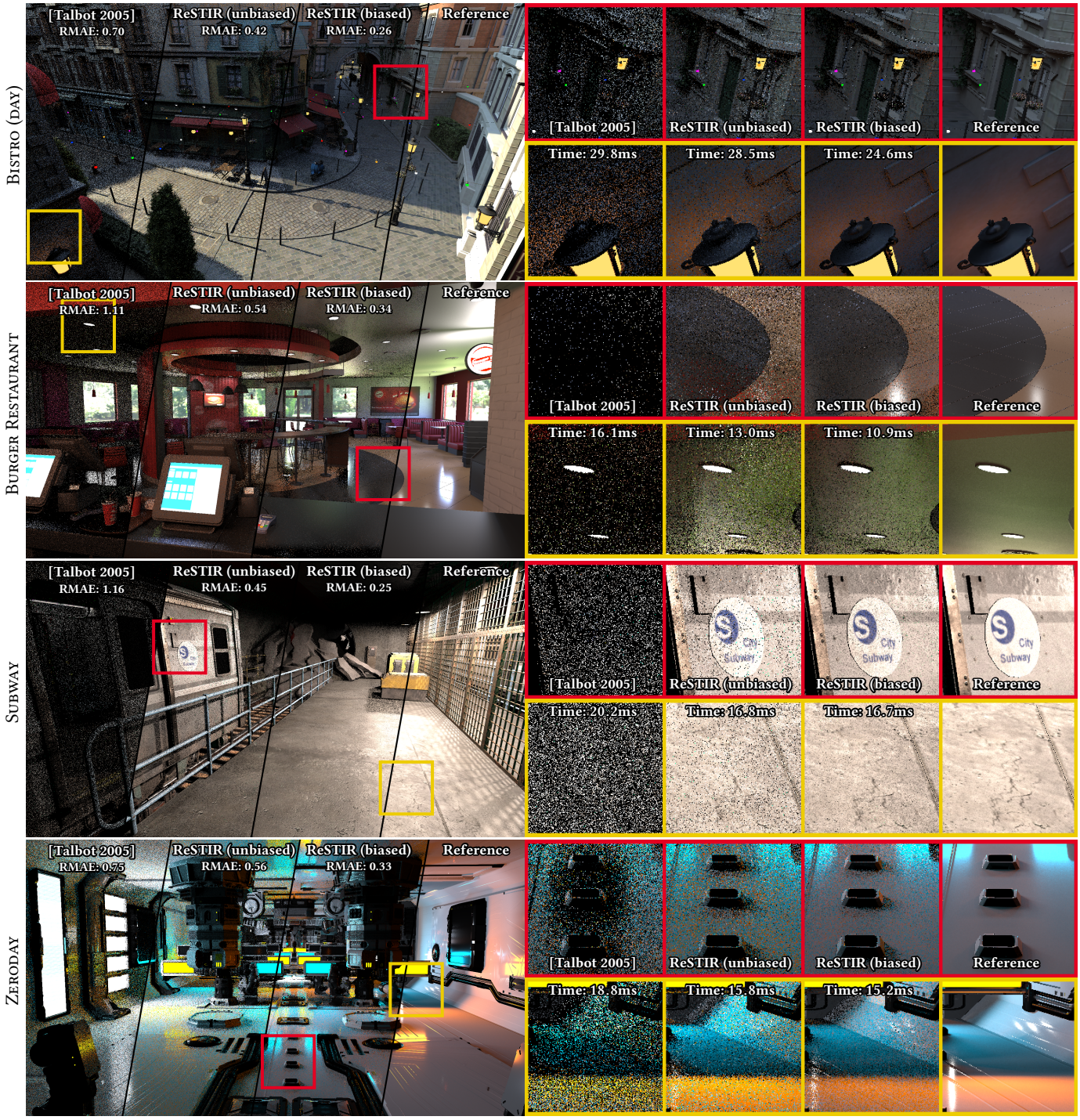


Fig. 8. Comparison of roughly equal-time renderings of a streaming implementation of Talbot et al. [2005] with our biased and unbiased spatiotemporal sample reuse. A converged reference is also shown for comparison. Bistro has 20,638 emissive triangles and an environment map, Burger Restaurant has 7,517 textured emissive triangles and a mostly-occluded environment map, Subway has 23,452 textured emissive triangles, and Zero Day animation has 10,973 dynamic emissive triangles. Bistro ©Amazon Lumberyard, Burger Restaurant ©Astuff, Subway ©silvertm, Zero Day ©beeple

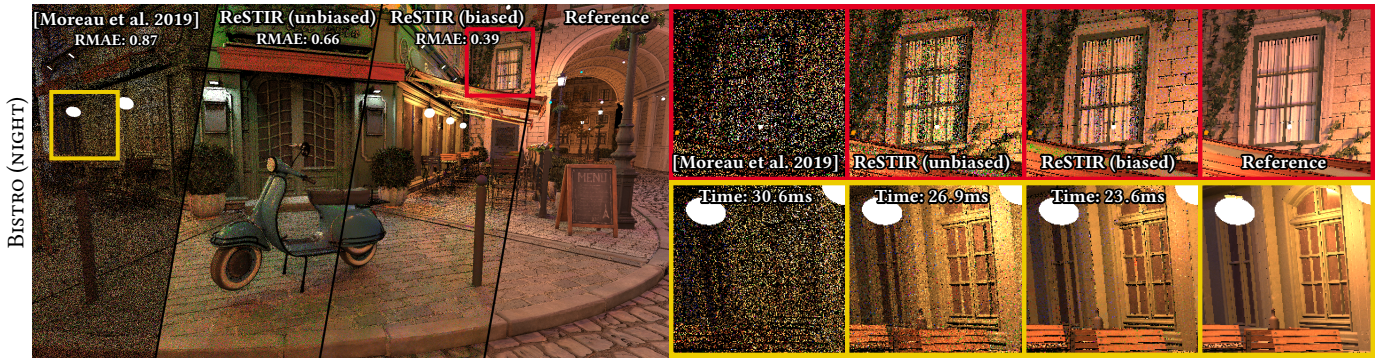


Fig. 9. An equal render time comparison of Moreau et al. [2019]'s light sampling scheme to our biased and unbiased sample reuse. Note our significant quality improvement, despite a simpler algorithm that requires no data structure updates for dynamic lights (not reported as part of their cost). The Bistro scene has 20,638 emissive triangles. Bistro ©Amazon Lumberyard

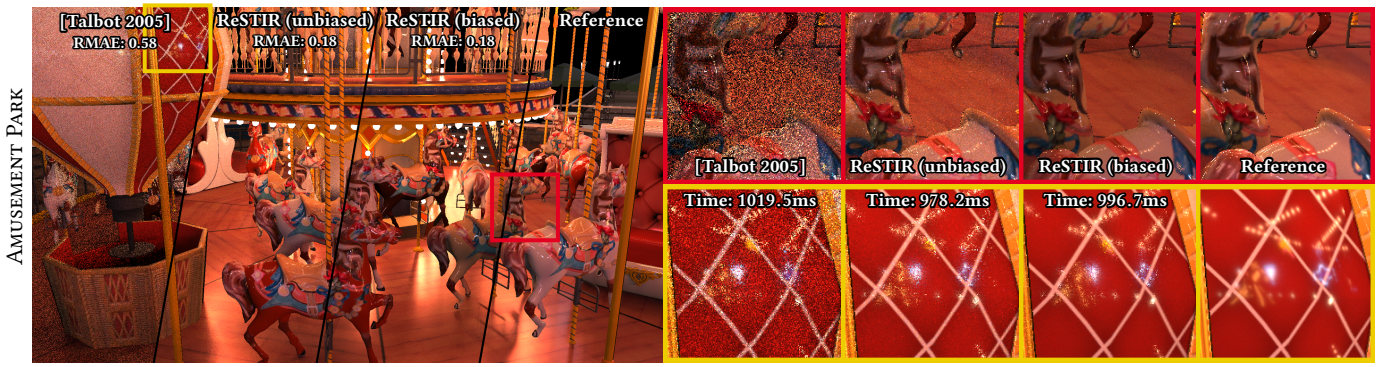


Fig. 10. An equal time comparison given a longer 1 s compute budget. We compare a streaming implementation of Talbot et al. [2005] with our biased and unbiased spatiotemporal sample reuse. Our Amusement Park scene has 3.4 million dynamic emissive triangles. Carousel ©carousel_world

rendering [Moreau et al. 2019], but because real-time renderers trace many fewer rays, the cost to construct and maintain these hierarchies is higher relative to the time spent rendering. Concurrent work by Lin and Yuksel [2020] uses a lower quality acceleration structure to lower the cost of maintaining the hierarchy, but still require data structure traversal and, in contrast to us, do not incorporate the BRDF. Our approach eliminates the cost of maintaining complex data structures and generates higher-quality light samples than light hierarchies by accounting for both the BSDF and lights' visibility.

Various other methods also adaptively construct PDFs for sampling direct lighting as part of rendering. Donikian et al. [2006] construct aggregate PDFs over fixed image blocks for light sampling in a progressive renderer. Their approach requires many rays to be traced in each pixel in order to find accurate PDFs. More recently, Vévoda et al. [2018] applied Bayesian online regression to create optimal light clusters. Their approach requires a prebuilt hierarchical Lightcut [Walter et al. 2005], which complicates application in scenes with dynamic lights. Neither of these accounts for the BSDF in the light sample. Related to these techniques are path guiding approaches [Hey and Purgathofer 2002; Jensen 1995; Müller et al. 2017; Vorba et al. 2014] that learn sampling PDFs for general illumination and can also be applied to direct lighting. None of these techniques have been shown to scale to real-time rates at low per-pixel sampling densities.

In interactive contexts, tiled shading [Olsson and Assarsson 2011] creates per-tile groups of important lights and accumulates per-pixel contributions only from these sources. While widely used commercially, these methods aim to reduce the number of lights affecting each pixel rather than efficiently aggregating all lighting. This biases the result, typically limiting each light's contribution to a limited area, though some stochastic variants [Tokuyoshi and Harada 2016] alleviate this bias.

Exploiting path reuse and spatial correlation. Reusing information between light-carrying paths has a long history in rendering. Algorithms based on virtual point lights (VPLs) generate numerous point-source emitters that approximate the illumination in an environment and then sample from them according to their expected contributions [Dachsbacher et al. 2014; Davidovič et al. 2010; Keller 1997; Ou and Pellacini 2011; Sbert et al. 2004; Segovia et al. 2006; Walter et al. 2006, 2005]. If sampled naively, VPLs require many rays per pixel for high-quality results. Alternatively, the cost of maintaining data structures for accurately sampling VPLs is challenging at real-time frame rates.

Another family of algorithms that reuse paths cache the incident illumination and interpolate it at nearby points; this approach is

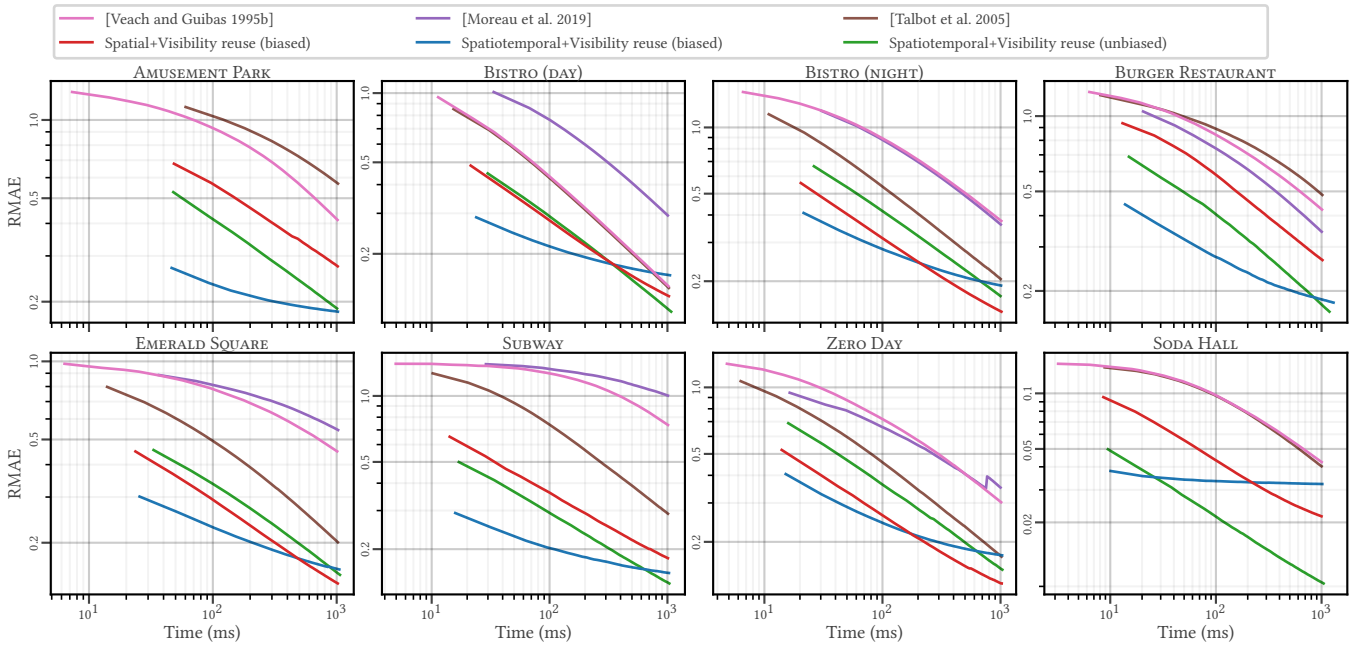


Fig. 11. The evolution of error (relative mean absolute error) in our scenes over render time. We compare Veach and Guibas-style MIS with lights sampled according to power, Moreau et al.'s light BVH, a streaming implementation of Talbot et al.'s RIS, and three variants of our algorithm: Biased and unbiased spatiotemporal and visibility reuse; as well as a biased form of spatial and visibility reuse, with no reliance on temporal information.

taken by both photon mapping [Deng et al. 2019; Jarosz et al. 2011, 2008b; Jensen 1996, 2001] and (ir)radiance caching [Jarosz et al. 2008a, 2012, 2008c; Křivánek et al. 2006, 2005; Schwarzhaupt et al. 2012; Ward and Heckbert 1992; Ward et al. 1988]. Those algorithms work well for slowly-changing illumination but struggle with rapid changes in visibility, as is often present with direct illumination.

Bidirectional path tracing reuses entire light carrying paths; early variants connected single vertices on pairs of camera and light subpaths, reusing their prefixes [Lafortune and Willems 1993; Veach and Guibas 1995a]. More recently, reusing paths enabled efficiency improvements and allows judicious choices of path connections [Chaitanya et al. 2018; Pajot et al. 2011; Popov et al. 2015; Tokuyoshi and Harada 2019]. Closely related is work on reusing paths in unidirectional light transport algorithms, where previously-sampled paths are stored and then connected to new paths [Bauszat et al. 2017; Bekaert et al. 2002; Castro et al. 2008; Xu and Sbert 2007]. Although these techniques can provide improved efficiency, a visibility ray must be traced each time a path is reused; in contrast, our method is able to reuse many more samples because it only traces rays for a small number of them.

Markov Chain Monte Carlo (MCMC) light transport algorithms [Cline et al. 2005; Hachisuka et al. 2014; Kelemen et al. 2002; Lai et al. 2007; Li et al. 2015; Otsu et al. 2018; Veach and Guibas 1997] reuse paths by maintaining one or more light-carrying paths and perturbing them so the distribution of weighted paths approximates the equilibrium radiance distribution in the scene. Efficiency is improved because these methods locally explore the space of valid light carrying paths. While often very effective at sampling challenging light-carrying paths, these algorithms require many samples per pixel before convergence and are often out-performed by traditional Monte Carlo techniques for typical light transport [Bitterli and Jarosz 2019]. Further, they suffer structured image artifacts due to correlation between samples.

All path reuse algorithms make trade-offs between efficiency gains and pixel correlations caused by path reuse. When reusing a path too often, artifacts can appear in rendered images. In general, the human visual system is more forgiving of high-frequency noise rather than structured artifacts [Cook 1986]. This has motivated work to distribute error as blue-noise across the image [Georgiev and Fajardo 2016; Heitz and Belcour 2019; Heitz et al. 2019]. While we exploit spatial correlation and extensive sample reuse across the image, our renderings contain high-frequency noise typical of uncorrelated Monte Carlo.

Resampling. Resampled importance sampling has various applications in rendering [Burke et al. 2004, 2005; Rubin 1987; Talbot 2005; Talbot et al. 2005]. Also related are sequential Monte Carlo (SMC) methods, where existing samples are perturbed and randomly accepted to approach a desired distribution [Ghosh et al. 2006; Pegoraro et al. 2008]. We build on RIS, transforming it into a streaming algorithm amenable to GPU implementation; ensuring it remains an unbiased estimator when sampling from different distributions; enabling spatiotemporal sample reuse; and incorporating multiple importance sampling.

Ratio & weighted estimators. Resampling techniques, including our method, are related to *ratio estimators*, which were originally used for sample surveys dating back to at least the 1950s. Similar estimators were independently developed in the Monte Carlo literature

under the name *weighted uniform sampling* (WUS) [Powell and Swann 1966], and applied to random walk problems by Spanier [1979] and Spanier and Maize [1994]. These were introduced to graphics by Bekaert et al. [2000] under the name *weighted importance sampling* (WIS) and later reintroduced by Stachowiak [2015] and Heitz et al. [2018] as ratio estimators. We detail WUS, WIS, and ratio estimators in Appendix B, but in essence, all three reduce variance by weighting (or taking a ratio of) each Monte Carlo sample with a chosen distribution correlated with the integrand.

In contrast, importance sampling (3), requires not only evaluating/weighting by the distribution, but also *generating* samples from this distribution. In their basic form, ratio estimators are biased, but are often preferred because they can result in lower variance while remaining consistent. Considerable work exists on making these estimators fully unbiased [Handscomb 1964; Hartley and Ross 1954; Mickey 1959; Rao and Beegle 1967; Worthley 1967], but to our knowledge, this topic has not yet been explored in graphics. In Appendix B we prove that WUS and WIS are just special cases of ratio estimators and that RIS [Talbot et al. 2005] can be viewed as a way to make these estimators unbiased.

(Weighted) reservoir sampling. Direct implementations of resampling-based sampling algorithms, such as RIS, typically require storing all candidate samples until one or more is selected. This is memory intensive, often prohibitively so for highly-parallel architectures such as GPUs. This challenge has been present for decades, in a variety of contexts. Generally, streaming algorithms often need stochastic selection from a list of unknown length. Reservoir sampling [Chao 1982; Vitter 1985] emerged in the early 1980s as a way to randomly select data stored on tape drives without random access, rewinding to reread, or storing it all in memory. Weighted variants allow selecting items with varying probability and have been applied in many domains (e.g., networking), with continuing research seeking to improve algorithmic complexity and statistical properties (e.g., Efraimidis [2015]; Efraimidis and Spirakis [2006]). While mostly unknown in graphics, the algorithm has recently been reinvented for stochastic order-independent transparency [Wyman 2016] and lighting from a hierarchy of VPLs [Lin and Yuksel 2019]. We use reservoir sampling in our streaming RIS algorithm, enabling a high-performance GPU implementation.

Denoising/reconstruction. Denoising and reconstruction frequently leverage path or sample reuse. While some approaches reconstruct from high-dimensional samples [Hachisuka et al. 2008; Lehtinen et al. 2011, 2012], most collapse these to 2D and rely on traditional image denoising filters, such as NL-means [Buades et al. 2005] or bilateral [Tomasi and Manduchi 1998], guided by auxiliary buffers to disambiguate MC noise from image features, often through some regression approach [Bitterli et al. 2016; Hachisuka et al. 2008; Kalantari et al. 2015; Lehtinen et al. 2011, 2013; Moon et al. 2014, 2015, 2016; Rousselle et al. 2016, 2011, 2012, 2013]. Zwicker et al. [2015]'s recent survey covers these in greater depth. Denoising has in large part enabled the transition to offline path tracing in movies [Christensen and Jarosz 2016] due to its ability to short-circuit the slow convergence tails of MC.

Work on interactive MC denoising has accelerated recently, exploring multi-scale [Dammertz et al. 2010], deep learning [Chaitanya et al. 2017; NVIDIA Research 2017], guided [Bauszat et al. 2011; He et al. 2010] spatio-temporal [Schied et al. 2017, 2018], and blockwise-regression filters [Koskela et al. 2019], in addition to sequences of filters [Mara et al. 2017]. These approaches are largely orthogonal to our work and can be applied to improve the output of our technique when not enough samples are taken for convergence (see Fig. 2).

8 CONCLUSION

We have introduced a new Monte Carlo approach to direct lighting based on a generalization of resampled importance sampling. It allows unbiased spatial and temporal reuse of nearby samples and leads to an even more efficient biased variant. Our algorithm delivers one to two orders of magnitude reduction in error compared to previous approaches while also requiring only simple image-space data structures. We have shown that it is suitable for high-performance GPU implementation, leading to real-time rendering of scenes with thousands and millions of dynamic light sources.

One way to view our technique is that we have shown that filtering and denoising need not remain a post-process that is performed once rendering completes—effectively, we have moved denoising into the core of the renderer and filter PDFs rather than colors. We see this as an important insight to spur future development of denoising algorithms, which have thus far remained specialized (and often carefully hand-tuned) postprocesses. It may also be worthwhile to develop new post-process denoising approaches that are adapted to the characteristics of the output of our algorithm or make use of unique features that it can provide, such as the individual candidate visibility values.

8.1 Limitations and Future Work

Similar to other algorithms relying on sample reuse, our method relies on exploiting correlations between pixels to improve image quality. When such opportunities are not available—e.g. near disocclusions, lighting discontinuities, high geometric complexity, fast moving lights—the quality of our method degrades and the noise reduction compared to the input samples is modest. While we generally saw our method performing better than prior work even in such challenging cases, making our method more robust to cases in which reuse is not possible is a fruitful direction for future work. Unlike post-processing methods such as denoising, our method still has the opportunity to trace additional samples, and it would be interesting to explore metrics that determine where our method fails, and allocate additional samples to those regions.

The main data structure of our algorithm consists of image buffers. While this makes our method fast, simple and memory efficient, it limits the use of our method to operations on the first vertex of the camera path (i.e. the primary hit point), and it cannot be easily extended to direct lighting or global illumination beyond the first hit. While direct lighting at the primary hit is an important problem in interactive applications, extending our algorithm beyond screen-space is an important area for future work. Of particular interest is applying our spatial and temporal resampling algorithm to a world-space data structure; algorithms such as path space hashing [Binder et al. 2019] may be useful in this context. Another possibility is to consider the combination of our resampling approach with path

reuse algorithms such as those developed Bekaert et al. [2002] and subsequent researchers.

Finally, although our GPU implementation targets interactive rendering, our algorithm applies equally to offline rendering. Temporal information may be unavailable when rendering a single still or parallelizing a sequence of frames over many computers, though additional rounds of spatial resampling with some visibility checks performed along the way would presumably give samples of similar quality to our spatiotemporal reuse. Furthermore, the granularity at which reservoirs are maintained merits investigation: pixel granularity is likely to be sub-optimal with complex geometry when image samples for a pixel intersect parts of the scene that are far away from each other, but the granularity of individual image samples may have a prohibitive memory cost. Clustering approaches that strike a balance between these two considerations may be effective.

ACKNOWLEDGMENTS

We thank Jacopo Pantaleoni for useful discussions during this project, and Jan Novák and Marco Salvi for their insightful feedback. This work was generously supported by a NVIDIA Research Professor Partnership and NSF grant #1844538.

REFERENCES

Pablo Bauszat, Martin Eisemann, and Marcus Magnor. 2011. Guided Image Filtering for Interactive High-Quality Global Illumination. *CGF* 30, 4 (June 2011), 1361–1368. https://doi.org/10/bwz228

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-Domain Path Reusing. *Proc. SIGGRAPH Asia* 36, 6 (Nov. 2017), 229:1–229:9. https://doi.org/10/gcqbjm

Philippe Bekaert, Mateu Sbert, and John Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Proc. EGWR*. Eurographics Association. https://doi.org/10/ggdwkn

Philippe Bekaert, Mateu Sbert, and Yves D. Willems. 2000. Weighted Importance Sampling Techniques for Monte Carlo Radiosity. In *Proc. EGWR*, B. Peroche and H. Rushmeier (Eds.). Springer-Verlag, 35–46. https://doi.org/10/ggdx9g

Nir Benty, Kai-Hwa Yao, Lucy Chen, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. 2019. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor

Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2019. Massively Parallel Path Space Filtering. *CoRR* abs/1902.05942 (2019). arXiv:1902.05942 http://arxiv.org/abs/1902.05942

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively Metropolised Monte Carlo Light Transport Simulation. *Proc. SIGGRAPH Asia* 38, 6 (Nov. 2019), 153:1–153:10. https://doi.org/10/dffp

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-Order Regression for Denoising Monte Carlo Renderings. *Proc. EGSR* 35, 4 (June 2016), 107–117. https://doi.org/10/f842kc

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2005. A Review of Image Denoising Algorithms, with a New One. *Multiscale Modeling & Simulation* 4, 2 (Jan. 2005), 490–530. https://doi.org/10/d4fhj8

David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. 2004. Bidirectional Importance Sampling for Illumination from Environment Maps. In *ACM SIGGRAPH Sketches*. 112. https://doi.org/10/b33qt2

David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. 2005. Bidirectional Importance Sampling for Direct Illumination. In *Proc. EGSR*. Eurographics Association, 147–156. https://doi.org/10/gfzsmz

Francesc Castro, Mateu Sbert, and John H. Halton. 2008. Efficient Reuse of Paths for Random Walk Radiosity. *Computers & Graphics* 32, 1 (Feb. 2008), 65–81. https://doi.org/10/dtkd67

Chakravarty R. Alla Chaitanya, Laurent Belcour, Toshiya Hachisuka, Simon Premoze, Jacopo Pantaleoni, and Derek Nowrouzezahrai. 2018. Matrix Bidirectional Path Tracing. In *Proc. EGSR (EI&I)*. Eurographics Association, Karlsruhe, Germany, 23–32. https://doi.org/10/ggfg6x

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *Proc. SIGGRAPH* 36, 4 (July 2017), 98:1–98:12. https://doi.org/10/gbxhcv

Min-Te Chao. 1982. A General Purpose Unequal Probability Sampling Plan. *Biometrika* 69, 3 (Dec. 1982), 653–656. https://doi.org/10/fd87zs

Per H. Christensen and Wojciech Jarosz. 2016. The Path to Path-Traced Movies. *Foundations and Trends® in Computer Graphics and Vision* 10, 2 (Oct. 2016), 103–175. https://doi.org/10/gfjwjc

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy Redistribution Path Tracing. *Proc. SIGGRAPH* 24, 3 (July 2005), 1186–1195. https://doi.org/10/b3xtrn

Robert L. Cook. 1986. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics* 5, 1 (Jan. 1986), 51–72. https://doi.org/10/cqwhcc

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable Realistic Rendering with Many-Light Methods. *CGF* 33, 1 (Feb. 2014), 88–104. https://doi.org/10/f5twgd

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. 2010. Edge-Avoiding À-Trous Wavelet Transform for Fast Global Illumination Filtering. In *Proc. HPG*. Eurographics Association, Saarbrucken, Germany, 67–75.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, Philipp Slusallek, and Kavita Bala. 2010. Combining Global and Local Virtual Lights for Detailed Glossy Illumination. *Proc. SIGGRAPH Asia* 29, 6 (Dec. 2010), 143:1–143:8. https://doi.org/10/bmktxb

Xi Deng, Shaojie Jiao, Benedikt Bitterli, and Wojciech Jarosz. 2019. Photon Surfaces for Robust, Unbiased Volumetric Density Estimation. *Proc. SIGGRAPH* 38, 4 (July 2019). https://doi.org/10.1145/3306346.3323041

Michael Donikian, Bruce Walter, Kavita Bala, Sebastian Fernandez, and Donald P. Greenberg. 2006. Accurate Direct Illumination Using Iterative Adaptive Sampling. *IEEE TVCG* 12, 3 (May 2006), 353–364. https://doi.org/10.1109/TVCG.2006.41

Pavlos S. Efraimidis. 2015. Weighted Random Sampling over Data Streams. (July 2015). arXiv:1012.0256

Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted Random Sampling with a Reservoir. *Inform. Process. Lett.* 97, 5 (March 2006), 181–185. https://doi.org/10/cw2qc4

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. the ACM on Computer Graphics and Interactive Techniques* 1, 2 (Aug. 2018), 25:1–25:17. https://doi.org/10/ggh89v

Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, Brian Green, Rob Pieké, Christopher Kulla, Christophe Hery, Ryusuke Villemin, Daniel Heckenberg, and André Mazzone. 2017. Path Tracing in Production (Parts 1 and 2). In *ACM SIGGRAPH Courses*. https://doi.org/10/gfz2ck

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH Talks*. ACM Press, Anaheim, California, 35:1–35:1. https://doi.org/10/gfznbx

Abhijeet Ghosh, Arnaud Doucet, and Wolfgang Heidrich. 2006. Sequential Sampling for Dynamic Environment Map Illumination. In *Proc. EGSR*, Tomas Akenine-Moeller and Wolfgang Heidrich (Eds.). Eurographics Association. https://doi.org/10/ggh89j

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. 2008. Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing. *Proc. SIGGRAPH* 27, 3 (Aug. 2008), 33:1–33:10. https://doi.org/10/fm6c2w

Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed Metropolis Light Transport. *Proc. SIGGRAPH* 33, 4 (July 2014), 100:1–100:10. https://doi.org/10/f6cswv

David C. Handscomb. 1964. Remarks on a Monte Carlo Integration Method. *Numer. Math.* 6, 1 (Dec. 1964), 261–268. https://doi.org/10/b6nf5f

Herman Otto Hartley and Arun Ross. 1954. Unbiased Ratio Estimators. *Nature* 174, 4423 (Aug. 1954), 270–271. https://doi.org/10/b4t29s

Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Guided Image Filtering. In *Proc. the European Conference on Computer Vision (ECCV)*. Springer-Verlag, Heraklion, Crete, Greece, 1–14.

E. Heitz and L. Belcour. 2019. Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds between Frames. *Proc. EGSR* 38, 4 (2019), 149–158. https://doi.org/10/ggjbxw

Eric Heitz, Laurent Belcour, V. Ostromoukhov, David Coeurjolly, and Jean-Claude Iehl. 2019. A Low-Discrepancy Sampler That Distributes Monte Carlo Errors as a Blue Noise in Screen Space. In *ACM SIGGRAPH Talks*. ACM Press, Los Angeles, California, 1–2. https://doi.org/10/ggjbxt

Eric Heitz, Stephen Hill, and Morgan McGuire. 2018. Combining Analytic Direct Illumination and Stochastic Shadows. In *Proc. I3D*. ACM Press, Montreal, Quebec, Canada, 2:1–2:11. https://doi.org/10/gfznb7

Heinrich Hey and Werner Purgathofer. 2002. Importance Sampling with Hemispherical Particle Footprints. In *Proc. SCCG*. ACM, Budmerice, Slovakia, 107–114. https://doi.org/10/fmx2jp

Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. 2008a. Radiance Caching for Participating Media. *ACM Transactions on Graphics* 27, 1 (March 2008), 7:1–7:11. https://doi.org/10/cwnw78

Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. 2011. A Comprehensive Theory of Volumetric Radiance Estimation Using Photon Points and Beams. *ACM Transactions on Graphics* 30, 1 (Jan. 2011), 5:1–5:19. https://doi.org/10/fcdh2f

Wojciech Jarosz, Volker Schönefeld, Leif Kobbelt, and Henrik Wann Jensen. 2012. Theory, Analysis and Applications of 2D Global Illumination. *ACM Transactions on Graphics* 31, 5 (Aug. 2012), 125:1–125:21. https://doi.org/10/gbbrkb

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. 2008b. The Beam Radiance Estimate for Volumetric Photon Mapping. *Proc. EG* 27, 2 (April 2008), 557–566. https://doi.org/10/bjsfsx

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. 2008c. Irradiance Gradients in the Presence of Participating Media and Occlusions. *Proc. EGSR* 27, 4 (June 2008), 1087–1096. https://doi.org/10/bg8nww

Henrik Wann Jensen. 1995. Importance Driven Path Tracing Using the Photon Map. In *Proc. EGWR*, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer-Verlag, 326–335. https://doi.org/10/gf2hcr

Henrik Wann Jensen. 1996. Global Illumination Using Photon Maps. In *Proc. EGWR*. Springer-Verlag, Vienna, 21–30. https://doi.org/10/fzc6t9

Henrik Wann Jensen. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Ltd., Natick, MA, USA.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *Proc. SIGGRAPH* 34, 4 (July 2015), 122:1–122:12. https://doi.org/10/f7mtzn

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *CGF* 21, 3 (Sept. 2002), 531–540. https://doi.org/10/bfrsqn

Alexander Keller. 1997. Instant Radiosity. In *Proc. SIGGRAPH*. ACM Press, 49–56. https://doi.org/10/fqch2z

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *Proc. SIGGRAPH* 38, 4 (July 2019), 37:1–37:14. https://doi.org/10/gf5jbj

Matias Koskela, Kalle Immonen, Markku Mäkitalo, Alessandro Foi, Timo Viitanen, Pekka Jääskeläinen, Heikki Kultala, and Jarmo Takala. 2019. Blockwise Multi-Order Feature Regression for Real-Time Path-Tracing Reconstruction. *Proc. SIGGRAPH* 38, 5 (June 2019), 138:1–138:14. https://doi.org/10/ggd8dj

Jaroslav Křivánek, Kadi Bouatouch, Sumanta N. Pattanaik, and Jiří Žára. 2006. Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping. In *Proc. EGSR*, Tomas Akenine-Möller and Wolfgang Heidrich (Eds.). Eurographics Association, Nicosia, Cyprus, 127–138. https://doi.org/10/gfzqhz

Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance Caching for Efficient Global Illumination Computation. *IEEE TVCG* 11, 5 (2005), 550–561. https://doi.org/10/csf2sw

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Proc. the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*, Vol. 93. Alvor, Portugal, 145–153.

Yu-Chi Lai, Shao Hua Fan, Stephen Chenney, and Charcle Dyer. 2007. Photorealistic Image Rendering with Population Monte Carlo Energy Redistribution. In *Proc. EGSR*. Eurographics Association, Grenoble, France, 287–295.

Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. 2011. Temporal Light Field Reconstruction for Rendering Distribution Effects. *Proc. SIGGRAPH* 30, 4 (July 2011), 1. https://doi.org/10/bpthww

Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. 2012. Reconstructing the Indirect Light Field for Global Illumination. *ACM Transactions on Graphics* 31, 4, Article 51 (July 2012), 10 pages. https://doi.org/10/gfzv9n

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. *Proc. SIGGRAPH* 32, 4 (July 2013), 95:1–95:12. https://doi.org/10/gbdghd

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics. *Proc. SIGGRAPH Asia* 34, 6 (Oct. 2015), 209:1–209:13. https://doi.org/10/f7wrcs

Daqi Lin and Cem Yuksel. 2019. Real-Time Rendering with Lighting Grid Hierarchy. *Proc. I3D* 2, 1 (June 2019), 8:1–8:17. https://doi.org/10/ggdzbp

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of I3D 2020)* 3, 1 (2020), 18. https://doi.org/10.1145/3384543

Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. 2017. An Efficient Denoising Algorithm for Global Illumination. In *Proc. HPG*. ACM Press, 3. https://doi.org/10/gfzndq

M. R. Mickey. 1959. Some Finite Population Unbiased Ratio and Regression Estimators. *J. Amer. Statist. Assoc.* 54, 287 (Sept. 1959), 594–612. https://doi.org/10/bqcrjk

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Transactions on Graphics* 33, 5 (Sept. 2014), 170:1–170:14. https://doi.org/10/f6km7m

Bochang Moon, Jose A. Iglesias-Guitian, Sung-Eui Yoon, and Kenny Mitchell. 2015. Adaptive Rendering with Linear Predictions. *Proc. SIGGRAPH* 34, 4 (July 2015), 121:1–121:11. https://doi.org/10/f7m2hp

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive Polynomial Rendering. *Proc. SIGGRAPH* 35, 4 (July 2016), 40:1–40:10. https://doi.org/10/f89mx6

Pierre Moreau, Matt Pharr, and Petrik Clarberg. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing. In *Proc. HPG*, Markus Steinberger and Tim Foley (Eds.). Eurographics Association. https://doi.org/10/ggh89m

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Proc. EGSR* 36, 4 (June 2017), 91–100. https://doi.org/10/gbnvrs

NVIDIA Research. 2017. NVIDIA® OptiX™ AI-Accelerated Denoiser. https://developer.nvidia.com/optix-denoiser

Ola Olsson and Ulf Assarsson. 2011. Tiled Shading. *JGGGT* 15, 4 (2011), 235–251. https://doi.org/10/bbfdms

Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. 2018. Geometry-Aware Metropolis Light Transport. *Proc. SIGGRAPH Asia* 37, 6 (2018), 278:1–278:11. https://doi.org/10/gf2r3t

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: Matrix Slice Sampling for the Many-Lights Problem. *Proc. SIGGRAPH Asia* 30, 6 (Dec. 2011), 179:1–179:8. https://doi.org/10/gfzm95

Anthony Pajot, Loïc Barthe, Mathias Paulin, and Pierre Poulin. 2011. Combinatorial Bidirectional Path-Tracing for Efficient Hybrid CPU/GPU Rendering. *Proc. EG* 30, 2 (2011), 315–324. https://doi.org/10/d6pbj2

Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A General Purpose Ray Tracing Engine. *Proc. SIGGRAPH* 29, 4 (July 2010), 66:1–66:13. https://doi.org/10/frf4mq

Vincent Pegoraro, Ingo Wald, and Steven G. Parker. 2008. Sequential Monte Carlo Adaptation in Low-Anisotropy Participating Media. *Proc. EGSR* 27, 4 (2008), 1097–1104. https://doi.org/10/fb55mk

Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. 2015. Probabilistic Connections for Bidirectional Path Tracing. *CGF* 34, 4 (2015), 75–86. https://doi.org/10/gfzwbh

Michael J. D. Powell and J. Swann. 1966. Weighted Uniform Sampling — a Monte Carlo Technique for Reducing Variance. *IMA Journal of Applied Mathematics* 2, 3 (Sept. 1966), 228–236. https://doi.org/10/bvgz69

J. N. K. Rao and LeNelle D. Beegle. 1967. A Monte Carlo Study of Some Ratio Estimators. *Sankhyā: The Indian Journal of Statistics, Series B (1960-2002)* 29, 1/2 (1967), 47–190. https://www.jstor.org/stable/25051590

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-Space Control Variates for Rendering. *Proc. SIGGRAPH Asia* 35, 6 (Nov. 2016), 169:1–169:12. https://doi.org/10/f9cphw

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2011. Adaptive Sampling and Reconstruction Using Greedy Error Minimization. *Proc. SIGGRAPH Asia* 30, 6 (Dec. 2011), 1. https://doi.org/10/c82v5c

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-Local Means Filtering. *Proc. SIGGRAPH Asia* 31, 6 (Nov. 2012), 195:1–195:11. https://doi.org/10/f96zx3

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising Using Feature and Color Information. *CGF (Proc. Pacific Graphics)* 32, 7 (Oct. 2013), 121–130. https://doi.org/10/gfzwbn

Donald B. Rubin. 1987. Comment. *J. Amer. Statist. Assoc.* 82, 398 (June 1987), 543–546. https://doi.org/10/gfzczq

Mateu Sbert, László Szécsi, and László Szirmay-Kalos. 2004. Real-Time Light Animation. *CGF* 23, 3 (2004), 291–299. https://doi.org/10/fksq8m

Christoph Schied. 2019. Video Series: Path Tracing for Quake II in Two Months. https://devblogs.nvidia.com/path-tracing-quake-ii/

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. In *Proc. HPG*. ACM, New York, NY, USA, 2:1–2:12. https://doi.org/10/ggd8dg

Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (Aug. 2018), 24:1–24:16. https://doi.org/10/ggd8dh

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-Based Error Control for Irradiance Caching. *Proc. SIGGRAPH Asia* 31, 6 (Nov. 2012), 1. https://doi.org/10/gbb6n4

Benjamin Segovia, Jean Claude Iehl, Richard Mitanchey, and Bernard Péroche. 2006. Bidirectional Instant Radiosity. In *Proc. EGSR*. Eurographics Association, 389–397.

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics* 15, 1 (Jan. 1996), 1–36. https://doi.org/10/ddgbgg

J. Spanier. 1979. A New Family of Estimators for Random Walk Problems. *IMA Journal of Applied Mathematics* 23, 1 (Jan. 1979), 1–31. https://doi.org/10/b8jdpn

Jerome Spanier and Earl H. Maize. 1994. Quasi-Random Methods for Estimating Integrals Using Relatively Small Samples. *SIAM Rev.* 36, 1 (1994), 18–44. https://doi.org/10/dxx9g9

Tomasz Stachowiak. 2015. Stochastic Screen-Space Reflections. In *Advances in Real-Time Rendering in Games, Part I (ACM SIGGRAPH Courses)*. https://doi.org/10/gf3s6n
Justin F. Talbot. 2005. *Importance Resampling for Global Illumination*. Masters Thesis. Brigham Young University. https://scholarsarchive.byu.edu/etd/663
Justin F. Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Proc. EGSR*. Eurographics Association, 139–146. https://doi.org/10/gfzsm2
Yusuke Tokuyoshi and Takahiro Harada. 2016. Stochastic Light Culling. *JCGT* 5, 1 (2016).
Yusuke Tokuyoshi and Takahiro Harada. 2019. Hierarchical Russian Roulette for Vertex Connections. *Proc. SIGGRAPH* 38, 4 (July 2019), 36:1–36:12. https://doi.org/10/gf5jbg
C. Tomasi and R. Manduchi. 1998. Bilateral Filtering for Gray and Color Images. In *Proc. the International Conference on Computer Vision (ICCV)*. 839–846. https://doi.org/10/dwsr88
Eric Veach and Leonidas J. Guibas. 1995a. Bidirectional Estimators for Light Transport. In *Proc. EGWR*. Springer-Verlag, 145–167. https://doi.org/10/gfznbh
Eric Veach and Leonidas J. Guibas. 1995b. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proc. SIGGRAPH*, Vol. 29. ACM Press, 419–428. https://doi.org/10/d7b6n4
Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *Proc. SIGGRAPH*, Vol. 31. ACM Press, 65–76. https://doi.org/10/bkjqj4
Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *Proc. SIGGRAPH* 37, 4 (July 2018), 125:1–125:12. https://doi.org/10/gd52ss
Jeffrey Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Software* 11, 1 (1985).
Thijs Vogels, Fabrice Rousselle, Brian Mcwilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with Kernel Prediction and Asymmetric Loss Functions. *Proc. SIGGRAPH* 37, 4 (July 2018), 124:1–124:15. https://doi.org/10/gd52sv
Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-Line Learning of Parametric Mixture Models for Light Transport Simulation. *Proc. SIGGRAPH* 33, 4 (Aug. 2014), 101:1–101:11. https://doi.org/10/f6c2cp
Alastair J Walker. 1974. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters* 10, 8 (1974), 127–128.
Bruce Walter, Adam Arbree, Kavita Bala, and Donald P Greenberg. 2006. Multidimensional Lightcuts. *Proc. SIGGRAPH* 25, 3 (July 2006), 1081–1088. https://doi.org/10/dzgsz7
Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. *Proc. SIGGRAPH* 24, 3 (Aug. 2005), 1098–1107. https://doi.org/10/dhp5d3
Gregory J. Ward. 1994. Adaptive Shadow Testing for Ray Tracing. In *Proc. EGWR (Focus on Computer Graphics)*, P. Brunet and F. W. Jansen (Eds.). Springer-Verlag, 11–20. https://doi.org/10/b7zrhm
Gregory J. Ward and Paul S. Heckbert. 1992. Irradiance Gradients. In *CE_EGWR93*, Alan Chalmers, Derek Paddon, and François X. Sillion (Eds.). Consolidation Express Bristol, Bristol, UK, 85–98.
Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. 1988. A Ray Tracing Solution for Diffuse Interreflection. *Proc. SIGGRAPH* 22, 4 (Aug. 1988), 85–92. https://doi.org/10/dk6rt5
Mike Winkelmann. 2015. Short Films by Beeple. https://www.beeple-crap.com/films
Reginald Gerald Worthley. 1967. *Unbiased Ratio-Type Estimators*. Masters Thesis. https://hdl.handle.net/2097/23084
Chris Wyman. 2016. Exploring and Expanding the Continuum of OIT Algorithms. In *Proc. HPG*. 1–11.
Chris Wyman, Shawn Hargreaves, Peter Shirley, and Colin Barré-Brisebois. 2018. Introduction to DirectX Raytracing. In *ACM SIGGRAPH Courses*. ACM Press, New York, NY, USA. https://doi.org/10/djqr
Qing Xu and Mateu Sbert. 2007. A New Way to Re-Using Paths. In *Computational Science and Its Applications – ICCSA 2007*, Osvaldo Gervasi and Marina L. Gavrilova (Eds.), Vol. 4706. Springer-Verlag, Berlin, Heidelberg, 741–750. https://doi.org/10/cggpq7
Cem Yuksel. 2019. Stochastic Lightcuts. In *Proc. HPG*. 27–32. https://doi.org/10.2312/hpg.20191192
Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (Proc. Eurographics State of the Art Reports)* 34, 2 (May 2015), 667–681. https://doi.org/10/f7k6kj

A EXPECTED RIS WEIGHT

Expanding Eq. (18) yields (the weight sums in the numerator and denominator cancel)

$$\frac{1}{p(y)}\sum_{i\in Z(y)}\int\cdots\int\frac{1}{\hat{p}(x_i)}\left[\frac{\cancel{\sum_{j=1}^{M}w_j(x_j)}}{M}\right]\left[\frac{w_i(x_i)}{\cancel{\sum_{j=1}^{M}w_j(x_j)}}\right]\left[\prod_{j=1}^{M}p_j(x_j)\right]\mathrm{d}x_1\ldots\mathrm{d}x_M. \tag{23}$$

Pulling all terms that do not depend on the integration variables outside, gives:

$$=\frac{1}{p(y)}\sum_{i\in Z(y)}\frac{p_i(x_i)}{\hat{p}(x_i)}\frac{w_i(x_i)}{M}\underbrace{\int\cdots\int\prod_{x_j\in\mathbf{x}\setminus x_i}p_j(x_j)\mathrm{d}x_1\ldots\mathrm{d}x_M}_{1}. \tag{24}$$

The remaining integral of all candidate PDFs (except x_i, which is fixed to be y), is simply 1. We can now simplify and use that $w_i(x)=\hat{p}(x)/p_i(x)$:

$$=\frac{1}{p(y)}\sum_{i\in Z(y)}\frac{p_i(x_i)}{\hat{p}(x_i)}\frac{w_i(x_i)}{M}=\frac{1}{p(y)}\sum_{i\in Z(y)}\frac{1}{M}=\frac{1}{p(y)}\frac{|Z(y)|}{M}. \tag{25}$$

B WEIGHTED, RATIO AND RESAMPLING ESTIMATORS

In contrast to importance sampling (3), which draws samples from some source PDF p, *weighted uniform sampling* (WUS) [Powell and Swann 1966] draws the samples x_i *uniformly*, and computes:

$$\langle L\rangle^N_{\text{wus}}=\sum_{i=1}^{N}f(x_i)\Bigg/\sum_{i=1}^{N}\hat{p}(x_i)\approx F, \tag{26}$$

where $\hat{p}(x)$ is a *normalized* PDF ideally correlated with f (but note that the samples x_i are generated uniformly).

Weighted importance sampling (WIS) [Bekaert et al. 2000] combines IS and WUS:

$$\langle L\rangle^N_{\text{wis}}=\sum_{i=1}^{N}\frac{f(x_i)}{\hat{p}(x_i)}w_i,\ \text{with}\ w_i=\frac{w(x_i)}{\sum_{j=1}^{N}w(x_j)},\ w(x)=\frac{\hat{p}(x)}{p(x)} \tag{27}$$

$$=\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\Bigg/\sum_{i=1}^{N}\frac{\hat{p}(x_i)}{p(x_i)}\approx F, \tag{28}$$

where the samples are drawn from a source PDF $p(x_i)$ that is easy to sample from (but only needs to be known up to a constant factor), and the target PDF $\hat{p}(x)$ can be a PDF for which no practical sampling algorithm exists as long as it is properly normalised. Weighted uniform sampling corresponds to the case where p is the constant PDF. Equation (27) is biased for finite values of N, but it is consistent, meaning that as $N\to\infty$, the bias and variance go to zero.

In *ratio estimation* [Hartley and Ross 1954; Heitz et al. 2018], the goal is to estimate the expected value $\bar{Y}$ of a random variable Y by leveraging a positively correlated random variable Z whose expectation $\bar{Z}$ is known. The classic, biased, ratio estimator drawns N sample pairs (y_i, z_i) and computes:

$$\langle\bar{Y}\rangle^N_{\text{rat}}=\bar{Z}\sum_{i=1}^{N}y_i\Bigg/\sum_{i=1}^{N}z_i\approx\bar{Y} \tag{29}$$

Equivalence of ratio estimation and WIS. If we define the random variables $Y=f(x)/p(x)$ and $Z=\hat{p}(x)/p(x)$, then WIS (28) can be written as

$$\langle L\rangle^N_{\text{wis}}=\sum_{i=1}^{N}y_i\Bigg/\sum_{i=1}^{N}z_i\ , \tag{30}$$

which is equivalent to the ratio estimator (29) since $\hat{p}$ is assumed normalized in WIS:

$$\bar{Z}=\int_D\frac{\hat{p}(x)}{p(x)}p(x)\,\mathrm{d}x=\int_D\hat{p}(x)\,\mathrm{d}x=1. \tag{31}$$

Relation of RIS to WIS. In WIS (27), consider either setting $N=1$, or for $N>1$ probabilistically evaluating only a single summand by selecting a single sample $y\in\{x_1,\ldots,x_N\}$ with probabilities dictated by w_i. The resulting one-sample WIS estimator becomes remarkably similar to RIS (6), which we restate for convenience:

$$\langle L\rangle^1_{\text{wis}}=\frac{f(y)}{\hat{p}(y)},\quad\text{whereas}\quad\langle L\rangle^{1,M}_{\text{ris}}=\frac{f(y)}{\hat{p}(y)}\cdot\left(\frac{1}{M}\sum_{j=1}^{M}w(x_j)\right). \tag{32}$$

Comparing these two estimators, we see that WIS is simply RIS without the average-of-weights term $\langle w\rangle^M\equiv\frac{1}{M}\sum_{j=1}^{M}w(x_j)=\frac{1}{M}\sum_{j=1}^{M}\hat{p}(x_j)/p(x_j)$. This is just an unbiased MC estimator of the target distribution's normalization factor in Eq. (31). Since we know that RIS (6) is unbiased, we know this factor acts as a bias-correction term.

In essence, by evaluating $f(y)/\hat{p}(y)$, RIS first forms a standard MC estimator (3) as if y came from the target distribution $\hat{p}$. For finite M, however, y is only approximately distributed with $\hat{p}$. RIS then uses $\langle w\rangle^M$ to correct for this approximate distribution and normalization of $\hat{p}$, and, critically, it does so using samples x_j that are *correlated* with $f(y)/\hat{p}(y)$. This correlated renormalization in RIS can be seen as a way to make WIS unbiased.