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.

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.


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

ACM Reference Format:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM. 0730-0301/2020/7-ART148 $15.00
https://doi.org/10.1145/3386569.3392481
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 $\omega$ due to direct lighting is given by an integral over all light emitting surfaces $A$:

$$L(y, \omega) = \int_A \rho(y, \frac{y-x}{d}) L_e(x \rightarrow y) G(x \leftrightarrow y) V(x \leftrightarrow y) \, dA_x,$$  

(1)

for BSDF $\rho$, 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 $\omega$ and shading point...
y for brevity and denoting differential area as dx, this simplifies to
\[ L = \int_A f(x) \, dx, \quad \text{where} \quad f(x) \equiv p(x) L_e(x) G(x) V(x). \] (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:
\[ \langle L \rangle_{IS}^N = \frac{1}{N} \sum_{i=1}^N f(x_i) / p(x_i) = L. \] (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 variability factor \( V(x) \). However, we can often draw samples proportional to individual terms in the integrand (e.g., the BSDF \( \rho \) 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:
\[ \langle L \rangle_{MIS}^{M,N} = \sum_{s=1}^M \frac{1}{N_s} \sum_{i=1}^{N_s} w_s(x_i) f(x_i) / p_s(x_i). \] (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, \( w_s(x) = N_s p(x) / \sum_j N_j p(x) \), is a popular and provably good choice [Veach and Guibas 1995b] for non-negative weights [Kondanapeni 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 \( x = \{x_1, \ldots, x_M\} \) from a source distribution \( \rho \) that is sub-optimal, but easy to sample from (e.g., \( \rho \propto L_e \)). It then randomly chooses an index \( z \in \{1, \ldots, M\} \) from this pool of candidates with discrete probabilities
\[ p(z \mid x) = \frac{w(x_z)}{\sum_{i=1}^M w(x_i)} \quad \text{with} \quad w(x) = \frac{\hat{p}(x)}{p(x)}, \] (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 ‘p’.) A sample \( y \equiv x_z \) is selected and used in the 1-sample RIS estimator:
\[ \langle L \rangle_{RIS}^{1,M} = \frac{f(y)}{\hat{p}(y)} \left( \frac{1}{M} \sum_{j=1}^M w(x_j) \right). \] (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:
\[ \langle L \rangle_{RIS}^{N,M} = \frac{1}{N} \sum_{i=1}^N \frac{f(y_i)}{\hat{p}(y_i)} \left( \frac{1}{M} \sum_{j=1}^M w(x_j) \right). \] (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 \( \sum_{i=1}^M w(x_i) \)

1. // Generate proposals \( x = \{x_1, \ldots, x_M\} \)
2. \( x \leftarrow \emptyset \)
3. \( w \leftarrow \emptyset \)
4. \( w_{\text{sum}} \leftarrow 0 \)
5. for \( i \leftarrow 1 \) to \( M \) do
6. \( \text{generate } x_i \sim p \)
7. \( x \leftarrow x \cup \{x_i\} \)
8. \( w_i \leftarrow \hat{p}_q(x_i) / p(x_i) \)
9. \( w_{\text{sum}} \leftarrow w_{\text{sum}} + w_i \)
10. \( w \leftarrow w \cup \{w_i\} \)
11. // Select from candidates \( x \)
12. Compute normalized CDF \( C \) from \( w \)
13. draw random index \( z \in [0, M) \) using \( C \) to sample \( \propto w_z \)
14. \( y \leftarrow x_z \)
15. return \( y, w_{\text{sum}} \)

Combining RIS with MIS. Above we assumed a single source PDF \( \rho \), 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 \( M \rightarrow \infty \) [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 \( \times \) 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.

```{r}
class Reservoir

1. y ← 0 // The output sample
2. wsum ← 0 // The sum of weights
3. M ← 0 // The number of samples seen so far

function update(x_i, w_i)

4. wsum ← wsum + w_i
5. M ← M + 1
6. if rand() < (w_i/wsum) then
7.   y ← x_i
8. return y

function reservoirSampling(S)

9. Reservoir r
10. for i ← 1 to M do
11.   r.update(S[i], weight(S[i]))
12. return r
```

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 \( w(x_i) \) such that \( x_i \) should be selected with probability

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

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

\[
\frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)},
\]

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

\[
\frac{w(x_i)}{\sum_{j=1}^{m} w(x_j)} \left(1 - \frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)}\right) = \frac{w(x_i)}{\sum_{j=1}^{m+1} w(x_j)},
\]

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 SIMULTERING 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 \( b(x) = p(x) L_e(x) G(x) \)
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

Algorithm 3: Streaming RIS using weighted reservoir sampling.

```plaintext
1 foreach pixel $q \in$ Image do
2   Image[i][q] ← shadePixel(RIS(q), q)
3 function RIS(q)
4   Reservoir r
5   for $i \leftarrow 1$ to $M$ do
6      generate $x_i \sim p$
7      r.update($x_i$, $\hat{p}_q(x_i)/p(x_i)$)
8      $r.W = \frac{1}{\hat{p}_q(r.q)} \cdot \left(\frac{1}{r.M} \cdot r.w_{sum}\right)$ // Equation (6)
9   return r
10 function shadePixel(Reservoir r, q)
11   return $f_q(r.q) \cdot r.W$
```

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 unshaded illumination ($\hat{p}(x) = \rho(x) L_r(x) G(x)$), 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 re-use 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 $\hat{p}_q(r.q)/\hat{p}_{q'}(r.q)$ to account for areas that were over- or undersampled at the neighbor compared to the current pixel. The resulting term $\hat{p}_q(r.q)/\hat{p}_{q'}(r.q) \cdot r.w_{sum}$ can be written more succinctly as $\hat{p}_q(r.q) \cdot r.W \cdot r.M$ 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
we render frames sequentially and feed forward their reservoirs, we store each pixel’s final reservoir for reuse in the next frame. If we render frames sequentially and feed forward their reservoirs, we store each pixel’s final reservoir for reuse in the next frame. If

Algorithm 4: Combining the streams of multiple reservoirs.

**Input**: Reservoirs $r_1$ to combine.

**Output**: A combined reservoir equivalent to the concatenated input streams of $r_1, \ldots, r_k$.

1. **function** combineReservoirs($q, r_1, r_2, \ldots, r_k$)
2.   Reservoir $s$
3.   foreach $r \in \{r_1, \ldots, r_k\}$ do
4.     $s$.update($r$)
5.     $s.M \leftarrow r_1.M + r_2.M + \ldots + r_k.M$
6.     $s.W = \frac{1}{s.M} \left( \frac{1}{s.W\text{sum}} \right)$ // Equation (6)
7.   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 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 $q$ for each pixel’s reservoir. If $q$ 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.

**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

1. **function** reservoirReuse(prevFrameReservoirs)
2.   reservoirs ← new Array[ImageSize]
3.   // Generate initial candidates
4.   foreach pixel $q \in$ Image do
5.     reservoirs[$q$] ← RIS($q$) // Alg. 3
6.   // Evaluate visibility for initial candidates
7.   foreach pixel $q \in$ Image do
8.     if shadowed(reservoirs[$q$], $q$) then
9.       reservoirs[$q$].$W \leftarrow 0$
10.  // Temporal reuse
11.  foreach pixel $q \in$ Image do
12.     $q' \leftarrow$ pickTemporalNeighbor($q$)
13.     reservoirs[$q$] ← combineReservoirs($q$, reservoirs[$q$.reservoirs[$q$].prevFrameReservoirs[$q'$]] // Alg. 4
14.  // Spatial reuse
15.  for iteration $i \leftarrow 1$ to $n$ do
16.    foreach pixel $q \in$ Image do
17.      $Q \leftarrow$ pickSpatialNeighbors($q$)
18.      $\mathbb{E} \leftarrow$ (reservoirs[$q'$].q $\in$ $Q$)
19.      reservoirs[$q$] ← combineReservoirs($q$, reservoirs[$q$.reservoirs[$q$, $\mathbb{E}$])
20.  // Compute pixel color
21.  foreach pixel $q \in$ Image do
22.    Image[$q$] ← shadePixel(reservoirs[$q$], $q$) // Alg. 3
23. return reservoirs

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 accommodate 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 accommodate our theory.

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

$$\langle I \rangle_{\text{ris}}^{1:M} = f(y) \cdot \frac{1}{p(y)} \frac{1}{M} \sum_{i=1}^{M} w(x_i) = f(y) W(x, z),$$  (11)

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

$$W(x, z) = \frac{1}{p(x)} \frac{1}{M} \sum_{i=1}^{M} w_i(x_i).$$  (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 $W(x, z)$ takes its place in Eq. (11). We can therefore guess that $W(x, z)$ must take on the role of the reciprocal PDF $1/p(y)$. However, $W(x, z)$ is a random variable: For a given output sample $y$ there are many $(x, z)$ that could have produced it, and which set of values (and therefore, which value for $W(x, z)$) is returned by RIS is random.

In order for Eq. (6) to be unbiased, the expected value of $W(x, z)$ 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 $p_q(r, y) \cdot r \cdot W \cdot r \cdot M$. We gave an intuitive justification of this weight in Section 3.2, but this term now has a straightforward explanation: $p_q(r, y) \cdot r \cdot W$ simply represents the standard RIS weight of $p_q(r, y)/p(r, y)$, except that we do not know the exact PDF $p(r, y)$ and use the estimate of the inverse PDF, $r \cdot W$ (Eq. (12)), instead. As $r \cdot y$ represents the result of combining multiple samples, the weight is additionally scaled by the number of candidates $r \cdot M$ that produced $r \cdot y$.

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 $x$ 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:

$$p(x) = \prod_{i=1}^{M} p_i(x_i).$$  (13)

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

$$p(z \mid x) = \frac{w_z(x_z)}{\sum_{i=1}^{M} w_i(x_i)} \quad \text{where} \quad w_i(x) = \frac{p_i(x)}{p(x)}.$$  (14)

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

$$p(x, z) = p(x) p(z \mid x) = \prod_{i=1}^{M} p_i(x_i) \frac{w_z(x_z)}{\sum_{i=1}^{M} w_i(x_i)}.$$  (15)

So what is $p(y)$? For a fixed output sample $y$, there are potentially many configurations of $x$ 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 $x_2, \ldots, x_M$ 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

$$Z(y) = \{i \mid 1 \leq i \leq M \land p_i(y) > 0\}.$$  (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 $y$:

$$p(y) = \sum_{i \in Z(y)} \int \cdots \int p(x^{\sim y}, i) dx_1 \cdots dx_M,$$  (17)

where $x^{\sim y} = \{x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_M\}$ is shorthand for the set of candidates with the $i$th 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 how the expected value of the RIS weight $W(x, z)$ 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 $W(x, z)$ only over those values of $x$ and $z$ for which $x_z = y$, and divide by $p(y)$: the probability density of the event $x_z = y$. This gives

$$\mathbb{E}_{x \sim y} [W(x, z)] = \sum_{i \in Z(y)} \int \cdots \int W(x^{\sim y}, i) p(x^{\sim y}, i) dx_1 \cdots dx_M,$$  (18)

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

In Appendix A we prove that this expression simplifies to:

$$\mathbb{E}_{x \sim y} [W(x, z)] = \frac{1}{p(y)} \frac{|Z(y)|}{M},$$  (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.
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 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 \( m(x_i) \):

\[
W(x, z) = \frac{1}{\hat{p}(x_i)} \left[ m(x_i) \sum_{i=1}^{M} w_i(x_i) \right].
\]

Repeating the derivation of the expected value of \( W \) shows that

\[
E_{x_i \sim y} [W(x, z)] = \frac{1}{\hat{p}(y)} \sum_{i \in Z(y)} m(x_i),
\]

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 \( m(x) = 1/\sum_i m(x_i) \). 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 \( p_2(x) \approx \max(2H(1/2 - x), 10^{-3}) \). 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 \( m(x_i) \) that sum to 1, for instance:

\[
m(x_i) = \frac{\hat{p}(x_i)}{\sum_{i=1}^{M} \hat{p}(x_i)},
\]

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 \( w_i(x) = \hat{p}(x)/\hat{p}_i(x) \) (Eq. (14)) as the weight, Talbot et al. use \( w_i(x) = \hat{p}(x)/\sum_i p_i(x) \). 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
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 \( \hat{p} \propto p \cdot L_{e} \cdot G \) 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 accommodate 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 $20 \times$ 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 prototype our method in the open-source Falc0 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.

![Image](image-url)

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
Amusement Park Bistro (night)

Vévoda et al. [2018] applied Bayesian online regression to create ACM Trans. Graph., Vol. 39, No. 4, Article 148. Publication date: July 2020.

illumination and can also be applied to direct lighting. None of et al. 2017; Vorba et al. 2014] that learn sampling PDFs for general 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 [Dachsacher et al. 2014; Davidović 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
Spatiotemporal reservoir resampling

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.


Bidirectional path tracing reuses entire light carrying paths; early variants connected single vertices on pairs of camera and light sub-paths, reusing their prefixes [LaFortune and Millems 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; Bekart 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; Pernaro 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 MIS.

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 et al. 2004].
and applied to random walk problems by Spanier [1979] and Spanier and Maize [1994]. These were introduced to graphics by Bekkaert 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 this 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. 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 reinvigorated 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 further 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 disconnections, 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
We thank Jacopo Pantaleoni for useful discussions during this project, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Chakravarty R. Alla Chaitanya, Laurent Belcour, Toshiya Hachisuka, Simon Premoze, David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. 2004. Bidirectional Importance
Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David
Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2011. Guided Image Filtering
REFERENCES
Partnership and NSF grant #1844538.
strike a balance between these two considerations may be effective.
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.

reuse algorithms such as those developed Bekta 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.

ACM Trans. Graph., Vol. 39, No. 4, Article 148. Publication date: July 2020.


A EXPECTED RIS WEIGHT

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

\[ \frac{1}{\mathbf{p}(y)} \sum_{i=1}^{N} \prod_{j=x_{i}}^{M} \frac{p_{i}(x_{j})}{\mathbf{p}(x_{j})} \left[ \int \ldots \int \frac{w_{i}(x_{j})}{\sum_{j=x_{i}}^{M} w_{i}(x_{j})} \prod_{j=x_{i}}^{M} p_{i}(x_{j}) \, dx_{j} \ldots \, dx_{M} \right]. \tag{23} \]

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

\[ \frac{1}{\mathbf{p}(y)} \sum_{i=1}^{N} \prod_{j=x_{i}}^{M} \frac{p_{i}(x_{j})}{\mathbf{p}(x_{j})} \left[ \int \ldots \int \prod_{j=x_{i}}^{M} p_{i}(x_{j}) \, dx_{j} \ldots \, dx_{M} \right]. \tag{24} \]

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 \( w_{i}(x) = \hat{p}(x)/\mathbf{p}(x) \):

\[ \frac{1}{\mathbf{p}(y)} \sum_{i=1}^{N} \prod_{j=x_{i}}^{M} \frac{p_{i}(x_{j})}{\mathbf{p}(x_{j})} \left[ \frac{1}{\mathbf{p}(y)} \sum_{i=1}^{N} 1^{M} = \frac{1}{\mathbf{p}(y)} |\mathbf{Z}(y)| \cdot \mathbf{M} \right]. \tag{25} \]

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:

\[ \langle L \rangle_{\text{wus}} = \sum_{i=1}^{N} f(x_{i})/\mathbf{p}(x_{i}) = F, \tag{26} \]

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:

\[ \langle L \rangle_{\text{w}} = \sum_{i=1}^{N} f(x_{i})/\mathbf{p}(x_{i}) \cdot w_{i}, \quad w_{i} = w(x_{i}) \approx \hat{p}(x)/\mathbf{p}(x), \tag{27} \]

\[ = \sum_{i=1}^{N} f(x_{i})/\mathbf{p}(x_{i}) \cdot \sum_{i=1}^{N} 1^{M} = F, \tag{28} \]

where the samples are drawn from a source PDF \( p(x) \) 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 \( \mathbf{p} \) 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 Rose 1954; Heitz et al. 2015], the goal is to estimate the expected value \( \hat{Y} \) of a random variable \( Y \) by leveraging a positively correlated random variable \( Z \) whose expectation \( \hat{Z} \) is known. The classic, biased, ratio estimator draws \( N \) sample pairs \((y_{i}, z_{i})\) and computes:

\[ \langle \hat{Y} \rangle_{\text{ris}} = \hat{Z} \sum_{i=1}^{N} y_{i} \sum_{i=1}^{N} z_{i} = \hat{Y}, \tag{29} \]

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

\[ \langle \hat{Y} \rangle_{\text{ris}} = \sum_{i=1}^{N} y_{i} \sum_{i=1}^{N} z_{i}, \tag{30} \]

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

\[ \hat{Z} = \int \hat{p}(x)/\mathbf{p}(x) \, dx = \int \hat{p}(x) \, dx = 1. \tag{31} \]

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 rewrite for convenience:

\[ \langle L \rangle_{\text{w}} = \int f(y)/\mathbf{p}(y) \, dy = \int \hat{p}(y)/\mathbf{p}(y) \, dy = 1. \tag{32} \]

Comparing these two estimators, we see that WIS is simply RIS without the average-of-weights term \( \langle w \rangle \mathbf{M} = \sum_{i=1}^{N} w_{i}(x_{i}) = \sum_{i=1}^{N} \hat{p}(x_{i})/\mathbf{p}(x_{i}) \mathbf{M} \). 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 \( f/\mathbf{p} \langle w \rangle \mathbf{M} \), 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 \( \langle w \rangle \mathbf{M} \) to correct for this approximate distribution and normalization of \( \hat{p} \), and, critically, it does so using samples \( x_{i} \) that are correlated with \( f/\mathbf{p} \langle w \rangle \mathbf{M} \). This correlated renormalization in RIS can be seen as a way to make WIS unbiased.

ACM Trans. Graph., Vol. 39, No. 4, Article 148. Publication date: July 2020.