Reservoir Splatting for Temporal Path Resampling and Motion Blur

JEFFREY LIU, University of Illinois Urbana-Champaign, USA
DAQI LIN, NVIDIA, USA
MARKUS KETTUNEN, NVIDIA, Finland
CHRIS WYMAN, NVIDIA, USA
RAVI RAMAMOORTHI, NVIDIA, USA and University of California San Diego, USA

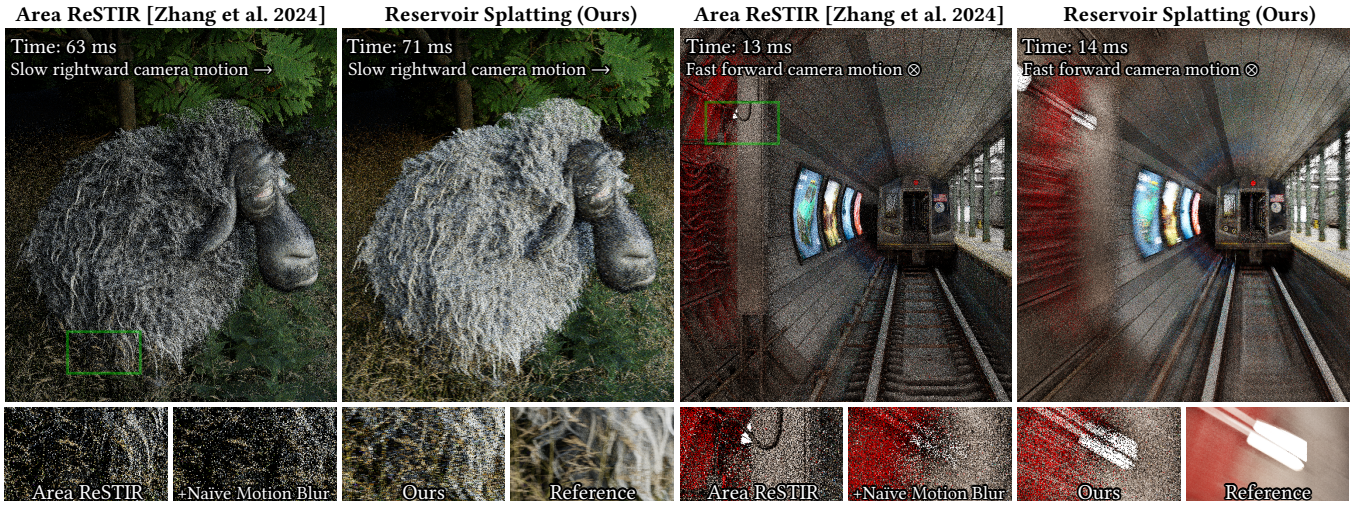


Fig. 1. *Prior ReSTIR methods [Zhang et al. 2024] degrade during camera motion, as sequential frames rarely shade identical primary hits, making perfect reuse tricky. This worsens near fine details, like foliage and fur, where sequential frames may not hit the same* surface. *By forward* splatting *hits from last frame, we guarantee they are reevaluated. Adding time to samples also enables resampling for motion blur while shading only one sample per pixel. Here we show two scenes under camera motion (Sheep In Forest and Subway) with stock Area ReSTIR [Zhang et al. 2024] and our new splatting-based ReSTIR. Insets also show an offline reference and a naïve motion blur baseline using Zhang et al. [2024] backprojection.*

Recent extensions to spatiotemporal path reuse, or ReSTIR, improve rendering efficiency in the presence of high-frequency content by augmenting path reservoirs to represent contributions over full pixel footprints. Still, if historical paths fail to contribute to future frames, these benefits disappear. Prior ReSTIR work *backprojects* to the prior frame to identify paths for reuse. Backprojection can fail to find relevant paths for many reasons, including moving cameras or subpixel geometry with differing motion.

We introduce *reservoir splatting* to reduce these failures. Splatting forward-projects the primary hits of prior-frame paths. Unlike backprojection, forward-projected path samples fall into the current-frame pixel relevant to their exact primary hits, making successful reuse more likely. This also enables motion blur for ReSTIR, by splatting at multiple time steps, and supports depth of field without the specialized shift maps needed previously.

Beyond enabling motion blur, splatting improves resampling quality over Zhang et al.'s [2024] Area ReSTIR at up to 10% lower cost. To improve robustness, we show how to MIS splatted and backprojected samples to help every current-frame pixel get at least one historical path proposed for reuse.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: real-time ray tracing, resampled importance sampling, ReSTIR, antialiasing, depth of field, motion blur

ACM Reference Format:
Jeffrey Liu, Daqi Lin, Markus Kettunen, Chris Wyman, and Ravi Ramamoorthi. 2025. Reservoir Splatting for Temporal Path Resampling and Motion Blur. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3721238.3730646

Authors' Contact Information: Jeffrey Liu, University of Illinois Urbana-Champaign, Urbana, USA, jliu179@illinois.edu; Daqi Lin, NVIDIA, Redmond, USA, daqil@nvidia.com; Markus Kettunen, NVIDIA, Helsinki, Finland, mkettunen@nvidia.com; Chris Wyman, NVIDIA, Redmond, USA, chris.wyman@acm.org; Ravi Ramamoorthi, NVIDIA, La Jolla, USA and University of California San Diego, La Jolla, USA, ravir@cs.ucsd.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGGRAPH Conference Papers '25, Vancouver, BC, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1540-2/25/08
https://doi.org/10.1145/3721238.3730646

1 Introduction

Recent importance sampling techniques based on resampled importance sampling (RIS) [Talbot et al. 2005] and iterated RIS (also known as ReSTIR) [Bitterli et al. 2020] dramatically improve rendering efficiency in interactive contexts by spatiotemporally reusing path samples. Over time, these methods can converge to near-optimal importance sampling with only one new sample per pixel (spp). This convergence relies on maintaining an easily-reusable path history.



Fig. 2. *Prior work, like Area ReSTIR [Zhang et al. 2024], converges quite well with 1 spp using a static camera. When moving, quality degrades significantly, especially for the high-frequency geometry in Emerald Square. This motivates our search for better ways to maintain historical samples.*

When history resets, quality temporarily degrades to that of naive, 1 spp path tracing. Reducing the frequency of spurious sample history resets is thus key to maintaining ReSTIR's efficiency.

Recently, Zhang et al. [2024] showed that elevating a path's dimensionality with the subpixel location allows better maintenance of temporal history in the presence of high-frequency normal maps. But like other ReSTIR papers [Lin et al. 2022; Ouyang et al. 2021], Zhang et al. still use temporal backprojection; this regularly changes a path's primary hit point between frames, whenever motion occurs. Such a history is challenging to reuse, especially where pixel color depends on aggregate geometry, e.g., hair or foliage; see Figure 2.

In this paper, we forward reproject (or *splat*) prior-frame samples to further improve reuse. Unless occluded in the current frame, such samples can be reused without changing their object-space primary hit; this stabilizes reuse and reduces history resets.

Our specific contributions in this paper include:

- A scatter-based reservoir reuse that *exactly* preserves object-space primary hits between frames (Section 4),
- A simple "backup" sample mechanism to fill holes between splatted samples, e.g., during zooming (Section 4.2),
- Applying splatting to motion blur; we show the first ReSTIR-accelerated motion blur algorithm (Section 4.4),
- Enabling depth of field without Zhang et al.'s [2024] specialized shift map or hand-tuned MIS weights (Section 4.5).

Overall, our work simplifies Area ReSTIR sampling, gives better performance, enhances resampling quality, and enables motion blur and depth of field out-of-the-box.

2 Related Work

Ray and path tracing [Kajiya 1986; Whitted 1979] have long proven costly, motivating researchers to explore techniques to optimize, reduce samples, and amortize costs. We overview a number of common methods, including temporal reprojection, denoising, gradient-domain rendering, path reuse, and resampling techniques. We then review algorithms specifically developed for motion blur rendering.

2.1 Temporal Reprojection and Denoising

Early real-time ray tracing researchers lacked the benefits of hardware acceleration [Kilgariff et al. 2018], often making do with significantly less than one ray per pixel. Frameless rendering [Bishop et al. 1994] recomputes a subset of pixels each frame, reusing others until they get updated. Adding prior-frame reprojection [Adelson and Hodges 1995; Corso et al. 2017] repositions reused pixels, and a render cache [Nehab et al. 2007; Scherzer et al. 2007; Walter et al. 1999] reuses computation over multiple frames. Adaptive frameless rendering [Dayal et al. 2005] carefully places new samples and runs (cached) samples through spatiotemporal reconstruction. Yang et al. [2011] reprojected bidirectionally, i.e., from both forward and backward frames, to help reconstruct intermediate frames.

Outside of ray tracing, games widely apply temporal antialiasing (TAA) [Yang et al. 2020] to reduce undersampling using prior-frame data. Modern TAA methods also upsample their output [Yang et al. 2009], but can introduce ghosting, blur, and shimmer. To reduce these artifacts, many apply fast neural networks [Liu 2020; Xiao et al. 2020] that extend and accelerate offline superresolution networks (e.g., Dong et al. [2015]).

Real-time denoisers (e.g., Schied et al. [2017, 2018]) often ingest low-sample inputs, so they employ temporal reprojection to increase effective sample counts [NVIDIA 2020b]. As in temporal superresolution, many denoisers also apply neural networks [Bako et al. 2017; Chaitanya et al. 2017; Işık et al. 2021], sometimes to uncover adaptive sampling opportunities [Hasselgren et al. 2020; Kuznetsov et al. 2018].

Our method is the first to demonstrate how prior-frame samples can be used via forward-reprojection without introducing bias.

2.2 Path Reuse

Path reuse [Bekaert et al. 2002] amortizes costs by reusing paths or segments within pixel blocks, though such reuse has challenges; Xu and Sbert [2007] explore ways to reduce tile correlations. Gradient-domain rendering builds correlated paths for close-by pixels with shift mappings [Kettunen et al. 2015; Lehtinen et al. 2013] to evaluate finite differences. Bauszat et al. [2017] use these shift mappings to better reuse specular paths within pixel blocks.

Recently, resampling algorithms [Bitterli et al. 2020] have significantly improved path reuse, demonstrating real-time performance for many light transport problems; our work fits into this category of spatiotemporal reservoir resampling (or ReSTIR) algorithms.

2.3 Resampling for Rendering

Resampled importance sampling (RIS) [Talbot et al. 2005] aggregates M samples, which are then resampled into N samples, approximately distributed according to a normalized target function. Bitterli et al.'s [2020] ReSTIR DI combines RIS (using $N = 1$) with weighted reservoir sampling [Chao 1982], with each per-pixel *reservoir* storing a sample whose distribution is refined by resampling from spatial and temporal neighbors. Ouyang et al. [2021] treat longer paths as virtual point lights [Keller 1997] to handle indirect light. Lin et al. [2022] shift full paths between neighbors to properly handle glossy surfaces. Their generalized RIS (GRIS) provides a mathematical foundation for ReSTIR. Zhang et al.'s [2024] Area ReSTIR improves robustness for depth of field and subpixel details by reusing lens and subpixel coordinates with fractional motion vectors. We build on Area ReSTIR, but redesign its temporal reuse.

2.4 Motion Blur

Motion blur arises as cameras have non-zero exposure times. Perceptually, it conveys relative motion within a frame. But temporal integration is expensive, so real-time methods use various approximations, e.g., accumulating multiple frames [Haeberli and Akeley 1990], blurring per-pixel [Rosado 2007], extruding geometry [Gribel et al. 2010; Tatarchuk et al. 2003], or screen-space velocity maps [McGuire et al. 2012]. See Navarro et al. [2011] for a survey. Such approximations typically favor performance over accuracy.

Ray tracers can stochastically sample time to render motion blur; this is generally more expensive but produces better results. Multi-dimensional adaptive sampling [Hachisuka et al. 2008; Meister and Hachisuka 2022] concentrates samples in motion-blurred regions. Frequency analysis [Egan et al. 2009] enables sparser sampling driven by a sheared space-time filter. Light field reconstruction [Lehtinen et al. 2011, 2012; Munkberg et al. 2014] can efficiently render motion blur. Covariance tracing [Belcour et al. 2013] computes a required per-pixel sample count, and reconstructs in image-space. Manzi et al. [2016] accelerate motion blur by evaluating temporal differences with cross-frame shift maps. Oberberger et al. [2022] further account for motion blur in the denoising process.

To our knowledge, we achieve the first motion blur via resampling, potentially enabling real-time motion blur in low-sample renderers.

3 Preliminaries

We first provide a review of key concepts related to ReSTIR.

3.1 Unbiased Contribution Weights

Since the result of RIS does not have a tractable PDF, Lin et al. [2022] abandon the traditional f/p estimators

$$\mathbb{E}\left[\frac{f(X)}{p(X)}\right] = \int_{\operatorname{supp} X} f(x)\,\mathrm{d}x \tag{1}$$

in favor of the more general $f(X)W_X$ estimators with

$$\mathbb{E}\left[f(X)W_X\right] = \int_{\operatorname{supp} X} f(x)\,\mathrm{d}x, \tag{2}$$

where random variable W_X is an *unbiased contribution weight* (UCW) for X. A random variable W_X is a UCW for X if and only if Equation 2 is true for all integrable f; this is equivalent to $\mathbb{E}\left[W_X|X\right] = 1/p_X(X)$ [Lin et al. 2022].

While RIS resampling generally results in intractable PDFs, it allows unbiased integration and iterative resampling with simple UCWs via this $f(X)W_X$ integration framework.

3.2 Shift Mappings

Reusing a path between pixels requires modifying some vertices to change which pixel it contributes to. Shift mappings T move paths between domains, e.g., pixels. Formally, a shift map from Ω_1 to Ω_2 is a bijective function from a subset of Ω_1 to a subset of Ω_2.

The *reconnection shift* [Lehtinen et al. 2013] is a commonly used shift mapping. Given two camera positions x_0 and y_0 and primary hits x_1 and y_1, this shift maps *base path* $\bar{x} = [x_0x_1x_2\dots x_n]$ into *offset path* $T(\bar{x}) = [y_0y_1x_2\dots x_n]$, reconnecting to $\bar{x}$ immediately after primary hit y_1. This works well on rough surfaces, but fails the goal of good shifts (that $f(\bar{x}) \approx f(T(\bar{x}))$) if any of x_1, x_2, or y_1 are nearly specular. Kettunen et al.'s [2015] *half-vector shift* and Lin et al.'s [2022] *hybrid shift* postpone reconnection until rough vertices are found, more effectively handling specular materials.

Forward and backward shifts T and T^{-1} need not be defined for all paths, but when defined, bijectivity is essential for unbiased path reuse[1]. Shift mappings also change path and probability densities, so formulas using shifts need Jacobian determinants $|T'(x)|$ or $|\partial T/\partial x|$.

3.3 GRIS

Given candidate samples $X_1, \dots, X_N$, their corresponding domains $\Omega_1, \dots, \Omega_N$, a target domain Ω for resampling, and a non-negative target function $\hat{p}$ defined in Ω, GRIS [Lin et al. 2022] aggregates $X_1, \dots, X_N$ into a result $Y \in \Omega$, approximately distributed proportional to $\hat{p}$. First, each $X_i \in \Omega_i$ is shifted to a similar sample $Y_i \in \Omega$ via shift map T_i:

$$Y_i = T_i(X_i). \tag{3}$$

Then, a resampling weight w_i is computed for each Y_i:

$$w_i = m_i(Y_i)\hat{p}(Y_i)W_{X_i}\left|\frac{\partial T_i}{\partial X_i}\right|, \tag{4}$$

where m_i is a resampling MIS weight, W_{X_i} is sample X_i's UCW in its original domain Ω_i, and $|\partial T_i/\partial X_i|$ is the Jacobian of shift T_i. Finally, output Y is resampled from the Y_i proportional to weights w_i. The new contribution weight for Y is

$$W_Y = \frac{1}{\hat{p}(Y)}\sum_{i=1}^{N} w_i. \tag{5}$$

If the supports of candidates Y_i together cover the support of $\hat{p}$, W_Y is unbiased, as per Section 3.1. Taking one *canonical sample* guarantees this coverage; a canonical X_c is sampled from $\Omega_c = \Omega$, with an identity shift map T_c, so that X_c alone covers $\hat{p}$'s support.

Lin et al. [2022] introduce the *generalized balance heuristic* to compute resampling MIS weights:

$$m_i(y) = \frac{c_i\,\hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{N} c_j\,\hat{p}_{\leftarrow j}(y)}, \tag{6}$$

which uses the "$\hat{p}$ from" function

$$\hat{p}_{\leftarrow j}(y) = \hat{p}_j(T_j^{-1}(y))\left|\frac{\partial T_j^{-1}}{\partial y}\right| \tag{7}$$

as an (unnormalized) proxy for the PDF of $y = T_j(z_j)$, originating in domain Ω_j as $z_j = T_j^{-1}(y)$. The Jacobian modifies the proxy the way probability densities transform in shift mappings. The *confidence weights* c_j control the relative weight of candidate samples.

3.4 ReSTIR

ReSTIR repeatedly applies GRIS to share samples spatiotemporally. Each pixel i stores a *reservoir* containing a sample X_i, its UCW W_{X_i}, and confidence weight c_i. See Wyman et al.'s [2023] course notes for details, but at a high level, the process typically goes as follows:

[1] Failure to ensure invertibility, i.e., $T(T^{-1}(y)) = y$ and $T^{-1}(T(x)) = x$, is a common source of bias.

Initial sampling. Each pixel i gets some number N_{init} *initial candidates*, e.g., newly-traced independent paths. One is selected via RIS, producing a canonical *initial sample* X_i^* with UCW $W_{X_i^*}$.

Temporal resampling. Each pixel i backprojects along its backward motion vector to choose prior-frame reservoir j. The results of GRIS between $Y_j = T_j(X_j)$ and X_i^*, using confidence weights c_j and N_{init}, replaces X_i, a new W_{X_i} is computed, and c_i is set to $\min(c_{\text{cap}}, c_j + N_{\text{init}})$, where c_{cap} is a fixed *confidence cap*.

Spatial resampling. Each pixel i chooses N_{spat} spatial neighbors $j_1, \ldots, j_{N_{\text{spat}}}$ randomly from a box around i, and performs GRIS between X_i and each X_{j_k}, using the corresponding confidence weights. The result overwrites X_i; W_{X_i} is updated with Equation 5, and c_i sums the samples' confidence weights, again clamped to c_{cap}.

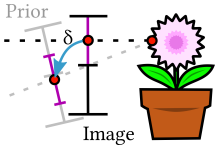
Shading. The final color for pixel i is evaluated as $f(X_i)W_{X_i}$, and the same process repeats in the next frame.

3.5 Area ReSTIR

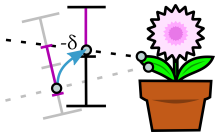
Prior to Area ReSTIR [Zhang et al. 2024], screen-space algorithms such as Lin et al. [2022] applied ReSTIR to path space *starting at secondary hits*, x_2, finding primary hits x_1 by ray tracing at predetermined subpixel locations (e.g., pixel centers).

Zhang et al. [2024] note this adds instabilities near high-frequency details: even if a reused path suffix $[x_2 x_3 \cdots x_n]$ remains valid, it may be incompatible with a new pixel's implicitly defined prefix $[x_0 x_1]$. To cure this, they add subpixel location (and lens position) as additional dimensions to reservoirs, essentially storing full paths $[x_0 x_1 \cdots x_n]$. Target functions no longer vary with each pixel's implicit prefix, as the shift maps propose complete, reusable paths. This significantly boosts reuse quality.

To improve subpixel precision for temporal path reuse, Zhang et al. [2024] backproject the hit point at each pixel i's center to the prior frame, via an image-space shift with motion vector δ_i, building a 1×1 pixel *fractional reservoir* around the result. This off-grid fractional reservoir is populated with samples from the overlapping 2×2 block of pixels (via RIS), and the chosen sample is shifted back to the current frame with motion vector $-\delta_i$, approximately retaining the primary hit.



As no samples in the 2×2 block of prior-frame pixels may fall in the fractional reservoir, Zhang et al.'s "fast" variant can leave this reservoir empty, producing excess noise. Prior to RIS, their "robust" variant first spatially shifts all samples in the 2×2 block to the fractional reservoir. This solves the issue but is more expensive.



Mathematically, Area ReSTIR integrates the measurement contribution function for each pixel i:

$$I_i = \int_{\Omega_i} f_i(\bar{x})\,\mathrm{d}\bar{x} = \int_{\Omega_i} h_i(\bar{x}) f(\bar{x})\,\mathrm{d}\bar{x},$$

where pixel i's path space $\Omega_i \subset \Omega$ is the set of paths $\bar{x}$ for which the pixel filter $h_i(\bar{x}) > 0$, and $f(\bar{x})$ is the path contribution. We use $\hat{p}_i(\bar{x})$ to denote the pixel-dependent target function for resampling, which we assume is defined as

$$\hat{p}_i(\bar{x}) = h_i(\bar{x})\hat{p}(\bar{x}), \tag{8}$$

often with $\hat{p} = f$, but cheaper approximations can also be used.

Note that Area ReSTIR only *approximately* retains primary hits; this may degrade temporal reuse near subpixel details. Our reservoir splatting *exactly* retains primary hits, improving temporal reuse.

3.6 Motion Blur

Zhang et al.'s Area ReSTIR integrates the measurement integral at a fixed time. However, rendering motion blur requires integrating over a time interval $[t_0, t_1)$, where $\Delta t = t_1 - t_0$ is the exposure time. Intensity of pixel i is then

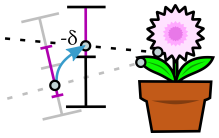
$$I_i = \int_{t_0}^{t_1} \int_{\Omega_i(t)} h_i(\bar{x}, t) f(\bar{x}, t)\,\mathrm{d}\bar{x}\,\mathrm{d}t, \tag{9}$$

where $h_i(\bar{x}, t)$ is the camera's pixel filter at time $t \in [t_0, t_1)$.

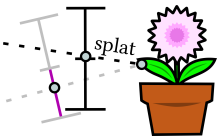
We experimented by naïvely extending Area ReSTIR for motion blur by adding sample time t to the reservoirs. But backprojecting each pixel's motion δ_i at a single time causes suboptimal reuse, as apparent motion depends on both sample time and subpixel location, which vary independently. Our splatting uses samples' real motion between frames, exactly contributing to the relevant pixels.

4 Reservoir Splatting

Prior ReSTIR methods *gather* reusable temporal neighbors in the prior frame; each pixel j is backprojected to the prior frame by a single backward motion vector δ_j to find candidates for reuse [Bitterli et al. 2020; Zhang et al. 2024]. Paths are then reused with the negated image-space motion $-\delta_j$, regardless of the reuse candidate's real image-space motion. This inexact tracking of shading points between frames can cause reuse failures in cases of subpixel detail, including factors like multiple objects (e.g., foliage) or region scaling (zooming), spuriously discarding the sample history that ReSTIR exploits.



We instead *scatter* prior-frame candidates by forward reprojection, or *splatting*, to map them to the current frame. Each splatted point uniquely maps to the current image, always contributing to the pixel it lands on, unless occluded. This exact mapping avoids discarding history unnecessarily.



Section 4.1 details how the math of resampling changes when splatting. Section 4.2 combines scatter- and gather-based reuse with appropriate MIS for better quality, albeit at higher cost. Section 4.3 discusses an approach to appropriately update reservoir confidence when splatting. Section 4.4 expands our sample splatting to real-time motion blur, and Section 4.5 describes how splatting also improves Zhang et al.'s [2024] depth of field.

4.1 GRIS with Scatter

Lin et al.'s [2022] GRIS theory requires defining input domains without looking at the samples. Prior work worked around this by reusing based on pixel-center motion vectors. But exactly preserving prior-frame samples' primary hits requires examining which

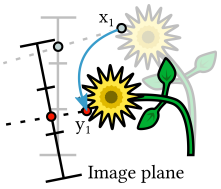
samples contribute to which pixel, which is not allowed. To reconcile this, we conceptually define *all* prior samples as contributing to *every* pixel, but ensure zero weight for those not splatting into current pixel j.[2] This allows applying GRIS for temporal resampling assuming:

(1) The target domain Ω is the current frame's full path space.
(2) The target function for pixel j is $\hat{p}_j(\bar{x}) = h_j(\bar{x})\hat{p}(\bar{x})$.[3]
(3) Input samples $X_1, \ldots, X_N$ come from *all* prior-frame area reservoirs, i.e., N is the screen size $H \times W$. Sample domains are $\Omega_i = \Omega^{\text{prev}}$, the entire prior frame's path space.
(4) Prior-frame candidates X_i use the same shift map $T_i = T$, where T is forward projection followed by the hybrid shift.
(5) Input X_{N+1} is a canonical sample $Y^* = Y^*_j$ for current pixel j. It uses identity shift T_{N+1}, as it lies in the current frame.

For efficient computation, we first initialize a GRIS reservoir with a canonical sample for each current-frame pixel. We then shift all prior paths X_i to the current frame as $Y_i = T(X_i)$, identifying the pixels each contributes to, and stream Y_i to these pixels' reservoirs (while avoiding race conditions).

While a balance heuristic (Section 4.1.2) for all-to-all reuse for all N pixels requires $O(N^3)$ total shifts, the splat operation implicitly zero-weights most prior-frame samples for a given pixel. This reduces to two shifts per pixel (forward-splatting the prior reservoir and reverse-splatting the initial sample) totaling $O(N)$ shifts.

4.1.1 Reprojection Shift Mapping. To splat path $\bar{x}$ into the current frame, we first transform its primary hit x_1 by keeping its object space position; given model-to-world transform M, we get $y_1 = M(M^{\text{prev}})^{-1}x_1$. For static scenes, $y_1 = x_1$. We then map lens vertex x_0 to the current frame. For pinhole cameras, y_0 is the new camera position. For other camera types, we can retain local lens coordinates or apply the cross-frame camera transform, i.e., $y_0 = M_v(M_v^{\text{prev}})^{-1}x_0$ given view-to-world matrix M_v. We test visibility between y_0 and y_1; if occluded, the shift fails and remains undefined. Finally, we shift path suffix $[x_2x_3 \ldots x_n]$ into $[y_2y_3 \ldots y_n]$ using any good shift, e.g., Lin et al.'s hybrid shift. Reprojection induces an additional Jacobian we must consider to avoid bias; we discuss this in Section 4.1.3.



4.1.2 GRIS Formulas. Section 4.1.1 defined resampling for current pixel j using $N+1$ inputs: N prior samples X_i shifted to $Y_i = T(X_i)$ in the current frame, plus a canonical sample $Y_{N+1} = Y^*$. In this subsection, we discuss how the box filter reduces the balance heuristic into simple expressions; the supplemental document contains derivations of MIS weights with general pixel filters.

Starting with prior sample X_i mapped to $Y_i = T_i(X_i)$, we substitute $\hat{p}_j(\bar{x}) = h_j(\bar{x})\hat{p}(\bar{x})$ into Equation 4. Using a box filter, $w_i = 0$ if Y_i is *not* in pixel j. Otherwise, for Y_i inside pixel j, $h_j(Y_i) = 1$, so we get

$$w_i = m_i(Y_i)\,\hat{p}(Y_i)W_{X_i}\left|\frac{\partial T}{\partial X_i}\right|. \tag{10}$$

The generalized balance heuristic [Lin et al. 2022] then reduces to

$$m_i(Y_i) = \frac{c_i\,\hat{p}^{\text{prev}}(X_i)\,|\partial T/\partial X_i|^{-1}}{c^*\,\hat{p}(Y_i) + c_i\,\hat{p}^{\text{prev}}(X_i)\,|\partial T/\partial X_i|^{-1}}, \tag{11}$$

given confidence weights c_i for samples X_i and c^* for canonical sample Y^*. This is because shifting Y_i into any prior-frame domain Ω_k results in the same path $T_k^{-1}(Y_i) = T^{-1}(Y_i) = X_i$, whose primary hit belongs only to pixel i's box filter. As a result, the balance heuristic in Equation 6 reduces from $N+1$ terms in the denominator to two in Equation 11. Scattering allows simply shifting each sample X_i once, contributing shifted sample Y_i to the pixel it falls in.

The initial sample $Y_{N+1} = Y^*$ gets resampling weight

$$w_{N+1} = m_{N+1}(Y^*)\,\hat{p}(Y^*)W_{Y^*}, \tag{12}$$

with

$$m_{N+1}(Y^*) = \frac{c^*\,\hat{p}(Y^*)}{c^*\,\hat{p}(Y^*) + c_r\,\hat{p}^{\text{prev}}(T^{-1}(Y^*))\left|\partial T^{-1}/\partial Y^*\right|}, \tag{13}$$

given confidence weight c_r at prior-frame pixel defined by the reverse splat $T^{-1}(Y^*)$. If the reverse splat fails due to occlusion or lying outside the image, that term becomes zero. Similarly, we only shift the canonical sample to the prior frame once, and at most one prior-frame pixel has positive box filter value for the splat $T^{-1}(Y^*)$. Thus, we again reduce from $N+1$ terms in the denominator to two.

4.1.3 Jacobian. Renderers often parametrize primary hits with subpixel locations. Consider splatting changing a previous-frame subpixel location v to the current frame u. In this parametrization, the Jacobian determinant from the reprojection (Section 4.1.1) is

$$\left|\frac{\partial u}{\partial v}\right| = \left|\frac{\partial u}{\partial y_1}\right|\left|\frac{\partial y_1}{\partial x_1}\right|\left|\frac{\partial x_1}{\partial v}\right|, \tag{14}$$

where $|\partial y_1/\partial x_1|$ accounts for object scaling and $|\partial y_1/\partial u|$ accounts for projection from world- to screen-space [Lehtinen et al. 2013]:

$$\left|\frac{\partial y_1}{\partial u}\right| = \left|\frac{\partial y_1}{\partial \omega}\right|\left|\frac{\partial \omega}{\partial u}\right| = \frac{\|y_1 - y_0\|^2}{\cos\theta_N}\cdot\cos^3\theta_V, \tag{15}$$

given angle θ_N between y_1's normal and $(y_0 - y_1)$, and angle θ_V between the camera's forward vector and $(y_1 - y_0)$. The full Jacobian $|\partial T/\partial\bar{x}|$ multiplies the subpixel Jacobian $|\partial u/\partial v|$ and the Jacobian of the remaining path, e.g., from the hybrid shift:

$$\left|\frac{\partial T}{\partial \bar{x}}\right| = \left(\frac{\cos\theta_N}{\cos\theta_N^{\text{prev}}}\frac{\cos^3\theta_V^{\text{prev}}}{\cos^3\theta_V}\frac{\|x_1 - x_0\|^2}{\|y_1 - y_0\|^2}\right)\cdot\left|\frac{\partial y_1}{\partial x_1}\right|\cdot\left|T'_{\text{hybrid}}\right|, \tag{16}$$

where $|\partial y_1/\partial x_1| = 1$ for rigid transformations. For non-rigid body deformations, this is computed as the ratio between the corresponding triangle's current and prior area.

4.1.4 Summary. We first scatter prior samples to the current frame, using them for resampling where they land. Next, we reverse-splat canonical samples to the prior frame to evaluate MIS weights. Last, we update UCWs for each pixels' chosen sample. Reservoir splatting remains unbiased as Jacobians compensate for any screen-space density change, and the initial samples ensure we fully cover the

[2] Similar formulations are used in light tracing, bidirectional path tracing, vertex connection and merging, and related techniques [Dutré et al. 1993; Georgiev et al. 2012; Lafortune and Willems 1993; Veach and Guibas 1995].
[3] Building on ideas from Area ReSTIR [Zhang et al. 2024], the target function is the measurement contribution function defined by Veach [1997].

path space. Efficiently implemented, reservoir splatting costs similar to Zhang et al.'s [2024] fast reuse in Area ReSTIR.

4.2 Backup Sample

Splatting projects to the current frame using the last frame's forward motion vectors; this leaves holes where no prior-frame pixel contributes. We can optionally fill holes with *backup samples*, e.g., backprojecting to the prior frame to find relevant samples.

To obtain a backup, we follow the *rounded* motion vector δ at each pixel center back to a prior-frame pixel b, whose sample X^b we give as an additional input to our temporal reuse. We directly use Zhang et al.'s [2024] proposed temporal shift T_b.

This simple method improves robustness, albeit at increased cost similar to Zhang et al.'s [2024] *robust* variant. Like any shift moving the shading point, this works best for low frequency geometry and lighting. In the supplementary material, we extend the MIS weights from Equations 11 and 13 to correctly account for backup samples.

4.3 Confidence Weight Update

Earlier methods set the confidence weight c_j to the capped sum of the inputs (Section 3.4). However, we conceptually have $H \times W$ inputs, of which only a few contribute. Setting the confidence weight based only on *contributing* reservoirs produces bias. Instead, we model confidence after Zhang et al. [2024], backprojecting the current pixel-center into the prior frame and bilinearly interpolating the confidence weights of the 2×2 overlapped pixels:

$$c_j = \min\left(c_i + \sum_{k'=1}^{4} \beta_{k'} c_{k'},\ c_{\text{cap}}\right). \tag{17}$$

Here, $\beta_{k'}$ is the bilinear weight corresponding to how much pixel k' overlaps with the backprojected pixel j. With the backup, we also add its confidence before clamping.

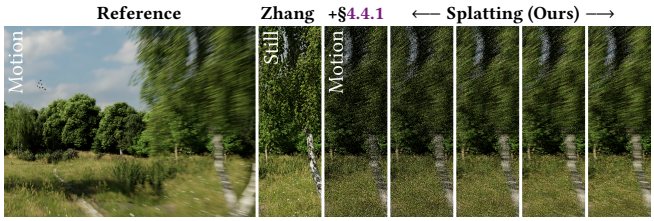
4.4 Motion Blur in ReSTIR

No prior spatiotemporal resampling algorithms handle motion blur, which requires integrating over a shutter time $[t_0, t_1)$ and associating a specific time t with each sample.

4.4.1 Naïve Area ReSTIR Extension. We started by naïvely extending Zhang et al. [2024], augmenting paths $\bar{x}$ into path-time pairs $(\bar{x}, t)$ and seeking a new shift between prior- and current-frame pairs. Shift mappings rely on invariants (e.g., a vertex or half-vector), and a sample's offset from the frame start is a possible invariant. A shift might preserve such an offset; given frame duration Δt, a sample time of t maps to $t \pm \Delta t$ in future or prior frames.

Backprojecting sample $(\bar{x}_j, t_j)$ from current pixel j to previous time $t_j - \Delta t$ defines a motion vector. But this represents only motion at $\bar{x}_j$, not all surfaces in pixel j. Area ReSTIR reuses from reservoirs with fixed motion. Even with a new time-offset shift, many paths in the reservoir may be irrelevant for the current pixel due to differing motion, sample time, or subpixel detail. This makes failed shifts and lost history more likely, degrading rendering quality; see Figure 3.

4.4.2 Splatting for Motion Blur. Comparatively, using scattering for motion blur is simple. Splatting forward-projects a *single* path, whereas backprojection seeks contributions for an *entire* pixel.



	Zhang	+§4.4.1	Splatting (Ours) 1x	2x	4x	8x
Splats per prior-frame sample	N/A	N/A	1x	2x	4x	8x
MAPE	0.333	0.872	0.857	0.709	0.642	0.612
FLIP	0.246	0.650	0.589	0.491	0.454	0.439
Time (ms)	53.2	48.2	47.5	55.5	60.1	77.6

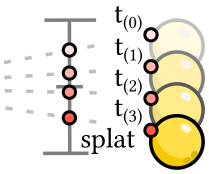
Fig. 3. *This* Landscape *has 4.3 billion triangles. For static cameras, Zhang et al. [2024] get good results, but the quality is lost in motion, even with naïve motion blur (Section 4.4.1). Splatting improves quality, lowers cost, and extends to multi-splatting to resample paths at multiple times during the exposure, further improving quality.*

Splatting by shifting forward in time Δt simply projects the prior primary hit to the correct point on its motion, mapping to a specific pixel. Given shift T in Section 4.1, our motion blur shift map $\hat{T}$ is:

$$(\bar{y}, t) = \hat{T}(\bar{x}, t^{\text{prev}}) = (T(\bar{x}), t^{\text{prev}} + \Delta t)\,, \tag{18}$$

which should be read as "forward project $\bar{x}$ to time $t^{\text{prev}} + \Delta t$", with T also implicitly depending on t.

4.4.3 Multi-Splatting for Motion Blur. Some motion blur methods [Haeberli and Akeley 1990] render repeatedly and accumulate. Similarly, splatting can repeatedly shift prior samples and splat into the current frame at different times. This reduces sample sparsity, improving reuse but at increased cost (see Figures 3 and 5).



To do this, we partition the shutter time $\tau = t_1 - t_0$ into K intervals of length τ/K, and splat each sample to all K intervals with shifts

$$(\bar{y}_{(n)}, t_{(n)}) = \hat{T}_n(\bar{x}, t^{\text{prev}}) = \left(T_n(\bar{x}), t_0 + \frac{n\tau}{K} + \frac{t^{\text{prev}} - t_0^{\text{prev}}}{K}\right), \tag{19}$$

where T_n reprojects $\bar{x}$ from t^{prev} to $t_{(N)}$, and $n = 0, \ldots, K-1$. As $\hat{T}_n$ squeezes the time dimension by $1/K$, we have $\left|\hat{T}_n'\right| = \left|T_n'\right|/K$ in resampling weight and MIS formulas.

4.5 Splatting for Depth of Field

Zhang et al. [2024] achieved real-time depth of field by reusing the subpixel location u. Shifts could then be defined by either reusing the lens position s (the *lens vertex copy* shift), or the primary hit x_1 (the *primary hit reconnection* shift). Primary hit reconnection improves reuse of bokeh samples for large apertures, but often leads to shift failures for small apertures. Neither shift produces acceptable results alone, so they must be combined via heuristic-based MIS weights.

In contrast, splatting explicitly preserves the primary hit x_1 and also allows reuse of a fixed lens sample s, while the image-space location u is simply defined by splatted sample location. This achieves high-quality bokeh using our single shift map. This saves compute time and simplifies the code. See Figure 4 for an illustration.

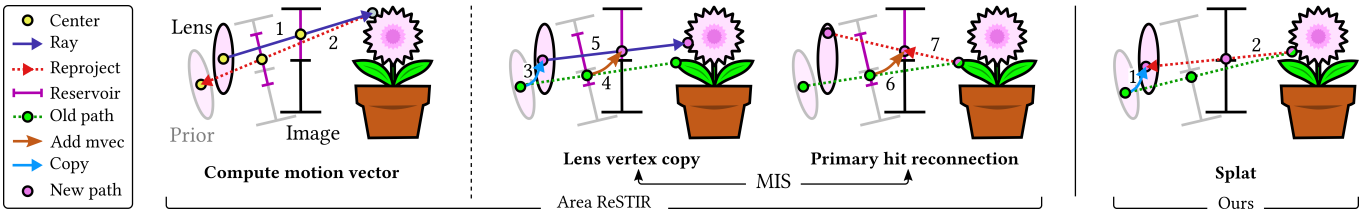


Fig. 4. *To remain robust, Area ReSTIR [Zhang et al. 2024] reuses temporally by one shift that preserves lens coordinates and another that preserves the primary hit, and merges with MIS. Our reservoir splatting alone preserves both.* ***Left****: Area ReSTIR computes a motion vector by tracing a primary hit through the current lens and pixel centers (1), then reprojecting towards the prior lens center onto the prior image plane (2).* ***Middle****: Lens vertex copy reuses prior path's lens coordinates (3), adds the motion vector to the image-space location (4), and traces a new primary hit (5). Primary hit reconnection adds the motion vector (6) and reprojects the old primary hit onto the lens through the image-space vertex (7).* ***Right****: Our reservoir splatting copies the local lens coordinates (1) and reprojects the old primary hit onto the image-space towards the new lens vertex (2).*

5 Implementation Details

We implement our reservoir splatting[4] in Falcor [Kallweit et al. 2022]. We use Zhang et al.'s [2024] area reservoir storage format, adding the explicit object-space primary hit, as well as a sample time when enabling motion blur. For correctness and efficiency, a few key details should be considered.

Multiple splats can land in any pixel, requiring atomics to avoid race conditions, as in order-independent transparency algorithms [Maule et al. 2011] such as real-time K-buffering [Bavoil et al. 2007].

Motion blur is tricky, as it traces rays at any time t. This requires arbitrary-time BVH queries. While recent hardware accelerates such *motion BVHs* [NVIDIA 2020,a], Falcor does not. Our prototype only handles blur from *dynamic* cameras in *static* scenes, allowing use of regular BVHs as geometry is time-invariant. Our algorithm should extend to dynamic scenes once we can trace rays at arbitrary times.

The supplemental material discusses these topics in further detail.

6 Results

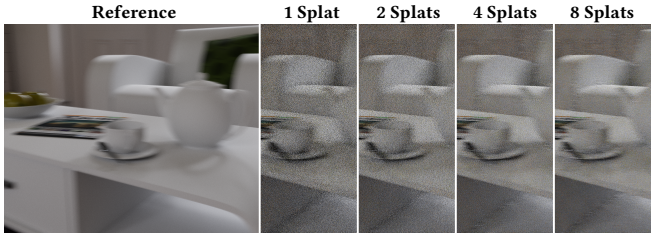
We generated all results at 1920×1080, with timings captured on an NVIDIA GeForce RTX 4090. We follow most settings from Zhang et al. [2024] and Lin et al. [2022], e.g., $c_{cap} = 20$, spatial neighbors from a 30 pixel radius. We enable Russian roulette, so average path depth is < 3 in most scenes. Unless stated, motion blur results use a simulated shutter of 1/24 seconds, regardless of frame time. We use MAPE to measure numerical errors and FLIP for perceptual errors [Andersson et al. 2021].

Figure 7 shows a cross-section of results, comparing our single splatting, with and without backup (Sections 4.4.2 and 4.4.3), to various baselines in tricky scenes.

The top three rows (Sheep In Forest, Hair, and Residential Lobby) compare with Zhang et al.'s [2024] Area ReSTIR, using their fast and robust temporal methods. Our splatting consistently runs 5-10% faster with 10-20% lower error than fast reuse, and splatting with a backup similarly outperforms robust reuse. In tricky cases, splatting alone also beats robust reuse; in simpler cases, e.g., along flat surfaces, splatting only gives quality on par with Zhang et al.'s fast reuse, though it usually retains a performance advantage.

The bottom rows (Bistro Exterior and Emerald Square) show motion blur, which was incompatible with ReSTIR before our work.

[4]Implementation available at https://github.com/Jebbly/Reservoir-Splatting.



	1 Splat	2 Splats	4 Splats	8 Splats
MAPE	0.600	0.444	0.346	0.290
FLIP	0.330	0.259	0.211	0.183
Time (ms)	14.5	18.0	22.8	31.9

Fig. 5. *The Living Room using multi-splatting with varying splats per prior pixel, with camera rotation generally leftwards.*

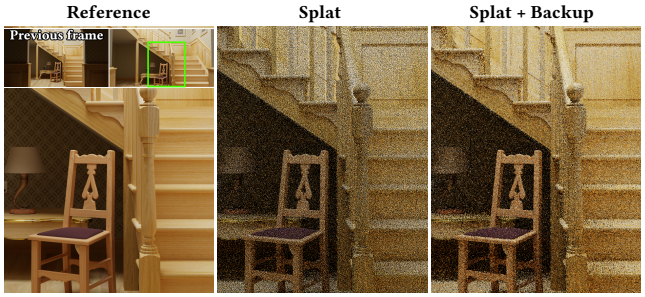


Fig. 6. *The Wooden Staircase captured under extreme forward motion. In such cases, splatting prior reservoirs into single pixels leaves holes between splats. Using a backup sample (Section 4.2) helps fill these holes, improving reuse.*

To avoid comparing to ad hoc post-process blur, we use a baseline of Zhang et al. [2024] augmented by our naïve, backprojected motion blur from Section 4.4.1. In this case, splatting again has 10-15% lower error and usually lower cost. From a quality perspective, Area ReSTIR's robust reuse has quite noticeable 2×2 pixel correlations. Such correlations are not present with splatting (alone), so even if the robust baseline has lower metrics, splatting often has less objectionable artifacts and runs twice as fast.

Qualitatively, multi-splatting could improve all our results in Figure 7 further (e.g., as in Figure 5), albeit at increased cost.

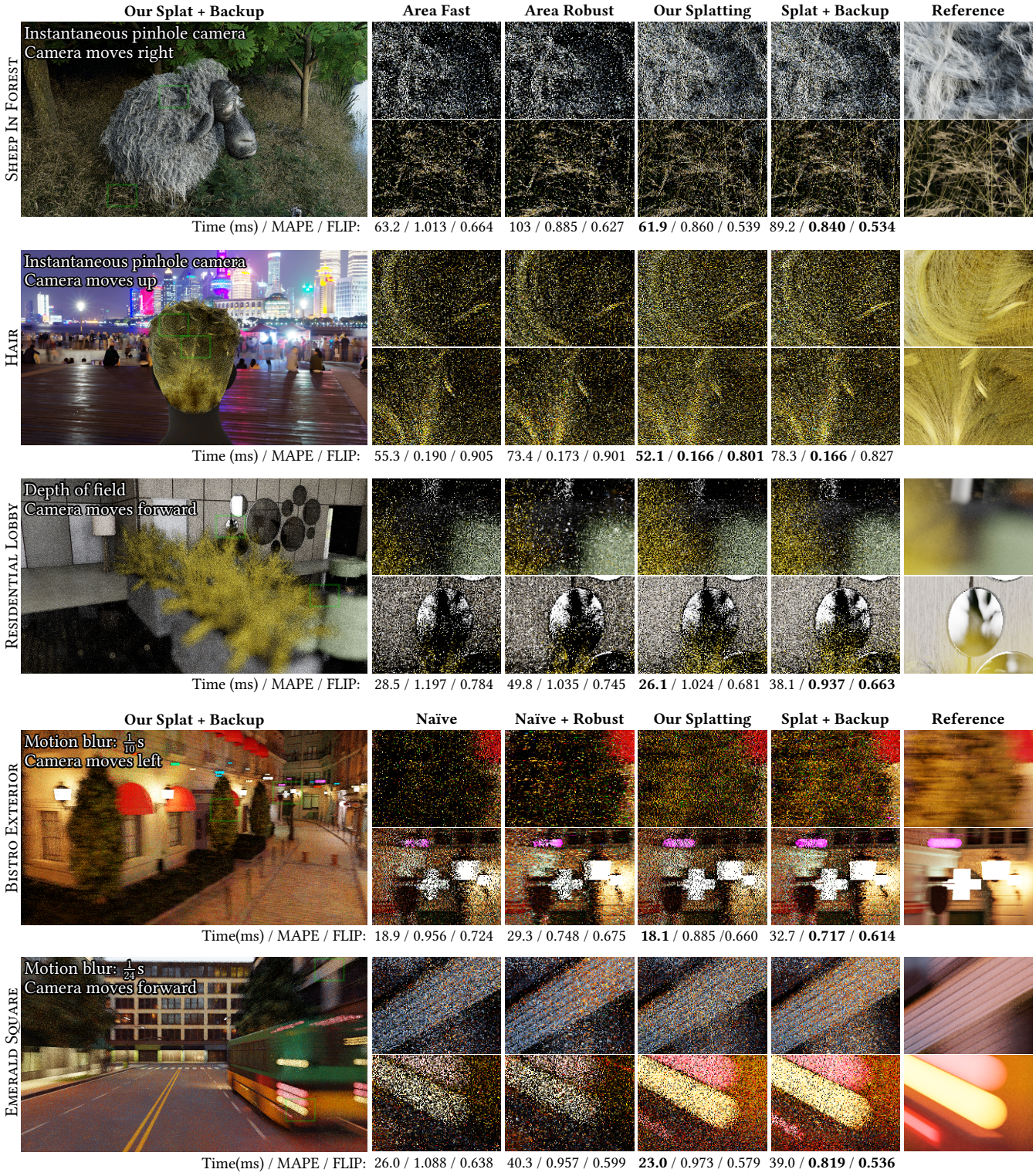


Fig. 7. *Here, we compare our work with various baselines. For Sheep In Forest, Hair, and Residential Lobby, we directly compare with Zhang et al.'s [2024] fast and robust variants. No prior resampling method quickly handles motion blur, so in Bistro Exterior and Emerald Square we compare against the naïve backprojected approach described in Section 4.4.1 (along with a* robust *variant similar to Zhang et al.).*

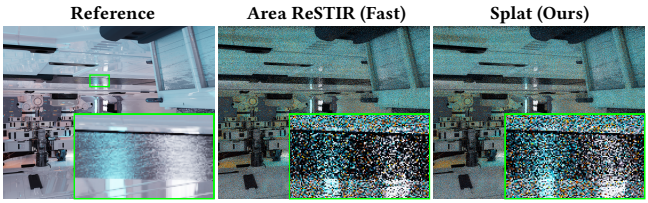


Fig. 8. *The Zero Day scene captured under moderate upward motion. Splatting improves high-frequency normal-mapped glossy materials compared to Area ReSTIR.*

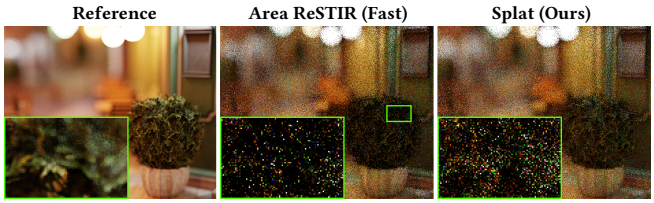


Fig. 9. *Captured under motion in the Bistro. Motion increases Area ReSTIR's [Zhang et al. 2024] noise due to less temporal reuse. Our splatting better preserves sample history, reducing noise, especially on foliage and specular highlights where correspondences between frames are more difficult to maintain.*



Fig. 10. *The Dining Room captured under slow rightward movement. Due to simple geometry and linear movement, Area ReSTIR's fractional reservoirs map well to the prior frame. Zhang et al.'s [2024] robust variant reuses from the 2 × 2 neighborhood, thus providing more effective samples than splatting plus a single backup sample.*

7 Discussion

Below we discuss some key takeaways from our work and results.

Resampled reuse deteriorates under motion. While ReSTIR converges impressively for static cameras, more frequent mismatches between temporal samples degrades the history under motion (see Figure 2). This is particularly apparent near high frequencies.

Gather versus scatter. This question repeatedly arises in graphics. Typically, similar algorithms can be designed with either. But a splat-based scatter *guarantees* testing every prior reservoir for relevance to a current pixel. Especially under motion, gathering via backprojected motion vectors makes no such guarantee. Quality bumps *just* from scattering are small in well-sampled scenes, but differences grow near high frequencies (e.g., high-frequency geometry in Figures 1, 3, 9, 7, and high-frequency material in Figure 8).

Splatting performance is fast and stable. Splatting performance depends on atomic contention. Even with fast motion, we averaged 0.6 to 0.9 valid prior-frame splats per pixel, with a maximum of around 5. Variations were well-distributed, leading to good GPU utilization. Despite the additional overheads discussed in Section 5, splatting only requires visibility queries; these are cheaper than the primary rays traced by other shifts that must find a new primary hit.

Backup samples are important for robustness. Splatting introduces holes, which grow when splats' relative motion diverges. Backup samples help reduce these holes, albeit at increased cost from the use *both* scattering and gathering. Figure 6 shows extremely fast motion, where these hole-filling benefits are clear. Interesting future work includes identifying backup reuse methods with smaller overheads. Even so, splatting with backup results in only six total shifts per pixel, while Zhang et al.'s [2024] robust reuse uses ten total shifts per pixel.

Multi-splatting. Multi-splatting gives another hole-reduction strategy, where the performance cost is relatively small, but can significantly improve motion blur quality (see Figure 3 and Figure 5). The Sheep in Forest and Subway in Figure 1 use 2× splatting.

Improving depth of field. Zhang et al. [2024] introduced the first resampling method for depth of field, improving quality significantly. However, they combined two shift maps with MIS, as neither shift handled samples in all circles of confusion. This adds variance, as one choice is nearly always better than the other.

Our splatting-based approach uses a single shift valid across the depth range, reducing noise near the focal plane and on high-frequency geometry. Additionally, due to the reduced shift count, our approach is routinely up to 10% faster (e.g., Figure 9 and Figure 7).

8 Limitations and Future Work

While splatting improves reuse for high-frequency details that *remain in image* frame-to-frame, it does not improve quality for *newly-disoccluded* surfaces; many of our tricky scenes have such pixels. While Zhang et al.'s [2024] and our backup samples can help fill such holes, better shifts or path mutations are likely needed for significant improvement (e.g., Kettunen et al. [2023]).

Extreme forward-motion may map one pixel to many. Backward-reprojection reuses a sample for many pixels with correlation; splatting reuses one-to-one, but may leave holes without backup samples. In simple cases where backprojected pixel footprints map well to the prior frame (e.g., Dining Room in Figure 10), splatting with backup may be slightly noisier than Area ReSTIR's robust reuse. This is because robust reuse pulls from a 2×2 neighborhood, which provides more samples but induces more correlation.

While multi-splatting also helps fill holes, it can spread fireflies present in reservoirs into multiple pixels along directions of movement (see Figure 3, far right). Such fireflies may cause problems for many modern denoisers, so mitigating them requires some thought.

Splatting follows a forward motion vector, so it inherits problems related to such methods; pixels representing geometry seen in a mirror will be splatted based on the mirror geometry, not the virtual

image. Our multi-splatting suggests we could follow *multiple* motion vectors per pixel to further improve the rendering of mirror-like surfaces, e.g., the clear-coated regions in Figure 8. This requires careful theoretical development.

9 Conclusion

We present a new method of forward projecting, or splatting, reservoirs between frames that improves reuse by reducing spuriously discarded sample histories. We show this works within Lin et al.'s [2022] generalized RIS theory, enabling splatting-based ReSTIR to remain unbiased.

Our work naturally applies to motion blur by repeatedly splatting samples for resampling. Furthermore, since we project exact primary hits, we also render depth of field with a single shift map, improving quality and performance over Zhang et al. [2024].

We thus believe splatting is the future way to implement ReSTIR for all scenarios. Enabling resampling for both scatter and gather operations enlarges the algorithmic toolbox, allowing ReSTIR to apply to a larger variety of problems, perhaps including better handling of mismatches between sampling density and screen size or combining with more complex bidirectional rendering techniques.

Acknowledgments

We thank Aaron Lefohn, Bill Dally, and Eric Shaffer for supporting this research. We also thank the anonymous reviewers for pointing out areas for improvement and clarifications.

The following environment maps were acquired from PolyHaven: Kloofendaal Sky for Dining Room, Spruit Dawn for Sheep in Forest, Shanghai Bund for Hair, Misty Pines for Residential Lobby, and Signal Hill Sunrise for Emerald Square. The Dining Room and Wooden Staircase scenes were acquired from Bitterli's [2016] rendering resources.

References

Stephen J Adelson and Larry F Hodges. 1995. Generating Exact Ray-Traced Animation Frames by Reprojection. *IEEE Computer Graphics and Applications* 15, 3 (1995), 43–52. doi:10.1109/38.376612

Pontus Andersson, Jim Nilsson, Peter Shirley, and Tomas Akenine-Möller. 2021. Visualizing Errors in Rendered High Dynamic Range Images. In *Conference of the European Association for Computer Graphics (Eurographics) Short Papers*. Eurographics-European Association for Computer Graphics. doi:10.2312/egs.20211015

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 97 (2017), 97:1–97:14 pages. doi:10.1145/3072959.3073708

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-Domain Path Reusing. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 229:1–229:9. doi:10.1145/3130800.3130886

Louis Bavoil, Steven Callahan, Aaron Lefohn, Joao Comba, and Claudio Silva. 2007. Multi-Fragment Effects on the GPU Using the k-Buffer. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 97–104. doi:10.1145/1230100.1230117

Philippe Bekaert, Mateu Sbert, and John H Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Eurographics Workshop on Rendering*. 125–134. doi:10.2312/EGWR/EGWR02/125-134

Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 2013. 5D Covariance Tracing for Efficient Defocus and Motion Blur. *ACM Trans. Graph.* 32, 3, Article 31 (July 2013), 18 pages. doi:10.1145/2487228.2487239

Gary Bishop, Henry Fuchs, Leonard McMillan, and Ellen J Scher Zagier. 1994. Frameless Rendering: Double Buffering Considered Harmful. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 175–176. doi:10.1145/192161.192195

Benedikt Bitterli. 2016. Rendering Resources. https://benedikt-bitterli.me/resources/ [Online; accessed 13-May-2025].

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron E. Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Transactions on Graphics (TOG)* 39, 4, Article 148 (Jul 2020), 17 pages. doi:10.1145/3386569.3392481

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. *ACM Transactions on Graphics (TOG)* 36, 4, Article 98 (jul 2017), 12 pages. doi:10.1145/3072959.3073601

Min-Te Chao. 1982. A General Purpose Unequal Probability Sampling Plan. *Biometrika* 69, 3 (1982), 653–656. doi:10.2307/2336002

Alessandro Dal Corso, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. 2017. Interactive Stable Ray Tracing. In *Proceedings of High Performance Graphics*. Article 1, 10 pages. doi:10.1145/3105762.3105769

Abhinav Dayal, Cliff Woolley, Benjamin Watson, and David Luebke. 2005. Adaptive Frameless Rendering. In *ACM SIGGRAPH 2005 Courses*. 24–es.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2015), 295–307. doi:10.1109/TPAMI.2015.2439281

Philip Dutré, Eric P. Lafortune, and Yves D. Willems. 1993. Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. In *3rd International Conference on Computational Graphics and Visualisation Techniques*. 128–137.

Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. 2009. Frequency Analysis and Sheared Reconstruction for Rendering Motion Blur. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 93, 13 pages. doi:10.1145/1576246.1531399

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. doi:10.1145/2366145.2366211

Carl Johan Gribel, Michael Doggett, and Tomas Akenine-Möller. 2010. Analytical Motion Blur Rasterization with Compression. In *High Performance Graphics*. 163–172. doi:/10.2312/EGGH/HPG10/163-172

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. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–10. doi:10.1145/1360612.1360632

Paul Haeberli and Kurt Akeley. 1990. The Accumulation Buffer: Hardware Support for High-Quality Rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*. 309–318. doi:10.1145/97879.97913

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. 2020. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum* (2020). doi:10.1111/cgf.13919

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo Denoising Using Affinity of Neural Features. *ACM Transactions on Graphics (TOG)* 40, 4, Article 37 (July 2021), 13 pages. doi:10.1145/3450626.3459793

James T Kajiya. 1986. The Rendering Equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150. doi:10.1145/15886.15902

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tomáš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor [Online; accessed 22-August-2023].

Alexander Keller. 1997. Instant Radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. 49–56. doi:10.1145/258734.258769

Markus Kettunen, Daqi Lin, Ravi Ramamoorthi, Thomas Bashford-Rogers, and Chris Wyman. 2023. Conditional Resampled Importance Sampling and ReSTIR. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11. doi:10.1145/3610548.3618245

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–13. doi:10.1145/2766997

Emmett Kilgariff, Henry Moreton, Nick Stam, and Brandon Bell. 2018. NVIDIA Turing Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/. [Online; accessed 9-December-2021].

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum* 37, 4 (2018), 35–44. doi:10.1111/cgf.13473

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Proceedings of Compugraphics*. 145–153. https://graphics.cs.kuleuven.be/publications/BDPT/

Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. 2011. Temporal Light Field Reconstruction for Rendering Distribution Effects. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 55, 12 pages.

doi:10.1145/1964921.1964950

Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. 2012. Reconstructing the Indirect Light Field for Global Illumination. *ACM Trans. Graph.* 31, 4, Article 51 (July 2012), 10 pages. doi:10.1145/2185520.2185547

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12. doi:10.1145/2461912.2461943

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 75:1–75:23. doi:10.1145/3528223.3530158

Edward Liu. 2020. DLSS 2.0 - Image Reconstruction for Real-Time Rendering with Deep Learning. GPU Technology Conference (GTC). https://developer.nvidia.com/gtc/2020/video/s22698-vid [Online; accessed 13-May-2025].

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. 2016. Temporal Gradient-Domain Path Tracing. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–9. doi:10.1145/2980179.2980256

Marilena Maule, João L. D. Comba, Rafael P. Torchelsen, and Rui Bastos. 2011. A Survey of Raster-Based Transparency Techniques. *Comput. Graph.* 35, 6 (Dec. 2011), 1023–1034. doi:10.1016/j.cag.2011.07.006

Morgan McGuire, Padraic Hennessy, Michael Bukowski, and Brian Osman. 2012. A Reconstruction Filter for Plausible Motion Blur. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Costa Mesa, California) *(I3D '12)*. Association for Computing Machinery, New York, NY, USA, 135–142. doi:10.1145/2159616.2159639

Daniel Meister and Toshiya Hachisuka. 2022. Lightweight Multidimensional Adaptive Sampling for GPU Ray Tracing. *Journal of Computer Graphics Techniques (JCGT)* 11, 3 (15 August 2022), 43–64. http://jcgt.org/published/0011/03/03/

Jacob Munkberg, Karthik Vaidyanathan, Jon Hasselgren, Petrik Clarberg, and Tomas Akenine-Möller. 2014. Layered Reconstruction for Defocus and Motion Blur. *Computer Graphics Forum* (2014). doi:10.1111/cgf.12415

Fernando Navarro, Francisco J. Serón, and Diego Gutierrez. 2011. Motion Blur Rendering: State of the Art. *Computer Graphics Forum* (2011). doi:10.1111/j.1467-8659.2010.01840.x

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the Symposium on Graphics Hardware*. 25–35. https://dl.acm.org/doi/10.5555/1280094.1280098

NVIDIA. 2020. GPU-Accelerated Motion Blur in Blender Cycles. https://www.nvidia.com/en-us/on-demand/session/gtcfall20-d22635/. [Online; accessed 19-January-2025].

NVIDIA. 2020a. NVIDIA Ampere GA102 GPU Architecture. https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf. [Online; accessed 22-January-2025].

NVIDIA. 2020b. NVIDIA Real-Time Denoisers (NRD). https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers. [Online; accessed 20-December-2024].

Max Oberberger, Matthäus G. Chajdas, and Rüdiger Westermann. 2022. Spatiotemporal Variance-Guided Filtering for Motion Blur. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 22 (2022). doi:10.1145/3543871

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Computer Graphics Forum* 40, 8 (2021), 17–29. doi:10.1111/cgf.14378

Gilberto Rosado. 2007. Motion Blur as a Post-Processing Effect. In *GPU Gems 3*, Hubert Nyugen (Ed.). Addison-Wesley Professional, 575–582.

Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. 2007. Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence. In *Proceedings of the Eurographics Symposium on Rendering*. doi:10.2312/EGWR/EGSR07/045-050

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 *Proceedings of High Performance Graphics*. 1–12. doi:10.1145/3105762.3105770

Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 24 (2018). doi:10.1145/3233301

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering*. 139–146. doi:10.2312/EGWR/EGSR05/139-146

Natalya Tatarchuk, Chris Brennan, and John Isidoro. 2003. Motion Blur Using Geometry and Shading Distortion. In *ShaderX2*, Wolfgang Engel (Ed.). Charles River Media.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulations*. Ph. D. Dissertation. Stanford University. https://graphics.stanford.edu/papers/veach_thesis/

Eric Veach and Leonidas Guibas. 1995. Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques*. Springer, 145–167. doi:10.1007/978-3-642-87825-1_11

Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive Rendering Using the Render Cache. In *Rendering Techniques' 99*. Springer Vienna, Vienna, 19–30.

Turner Whitted. 1979. An Improved Illumination Model for Shaded Display. In *Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques*. 14. doi:10.1145/800249.807419

Chris Wyman, Markus Kettunen, Daqi Lin, Benedikt Bitterli, Cem Yuksel, Wojciech Jarosz, Pawel Kozlowski, and Giovanni De Francesco. 2023. A Gentle Introduction to ReSTIR: Path Reuse in Real-Time. In *ACM SIGGRAPH 2023 Courses* (Los Angeles, California). doi:10.1145/3587423.3595511

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-Time Rendering. *ACM Transactions on Graphics (TOG)* 39, 4, Article 142 (Aug 2020), 12 pages. doi:10.1145/3386569.3392376

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.). 741–750. doi:10.1007/978-3-540-74477-1_67

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. doi:10.1111/cgf.14018

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized Supersampling. *ACM Transactions on Graphics (TOG)* 28, 5 (Dec. 2009), 1–12. doi:10.1145/1618452.1618481

Lei Yang, Yu-Chiu Tse, Pedro V. Sander, Jason Lawrence, Diego Nehab, Hugues Hoppe, and Clara L. Wilkins. 2011. Image-Based Bidirectional Scene Reprojection. *ACM Trans. Graph.* (Dec. 2011), 1–10. doi:10.1145/2070781.2024184

Song Zhang, Daqi Lin, Markus Kettunen, Cem Yuksel, and Chris Wyman. 2024. Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–13. doi:10.1145/3658210