Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing

SONG ZHANG*, University of Utah, USA
DAQI LIN*, NVIDIA, USA
MARKUS KETTUNEN, NVIDIA, Finland
CEM YUKSEL, University of Utah, USA
CHRIS WYMAN, NVIDIA, USA

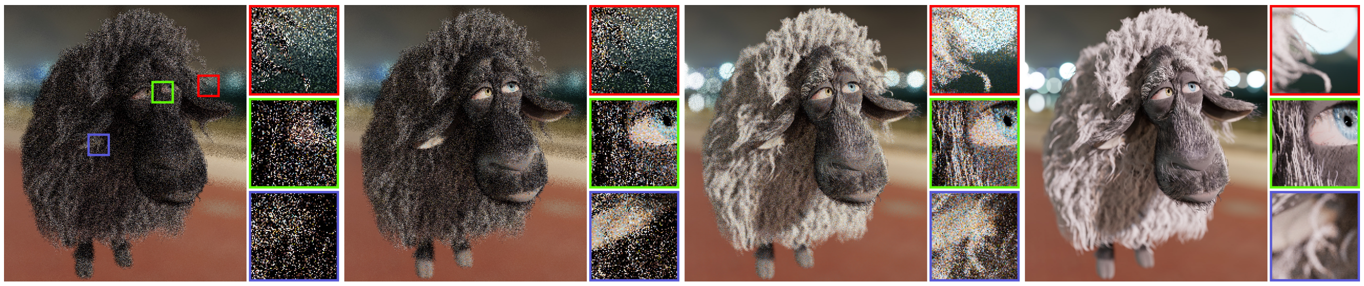

(a) Path tracing (1spp), 4.2 ms
(b) ReSTIR PT, 9.5 ms
(c) Area ReSTIR (Ours), 14.0 ms
(d) Ground Truth

Fig. 1. *Prior ReSTIR methods [Bitterli et al. 2020] reuse paths between fixed subpixel locations. Randomizing paths' subpixel locations avoids aliasing, but changes their primary hits. In pixels with high-frequency content, the changing normals and occlusions often make the new primary hits incompatible with later vertices (from reused, shifted paths), preventing effective reuse. Here the* Franck *model, with complex fur, is lit by the* Blue Lagoon *HDR environment. (a) Path tracing has high variance at low sample rates. On smoother surfaces, (b) ReSTIR PT [Lin et al. 2022] reuses samples for large quality improvements while shading only one path per pixel. But reusing samples in a higher-dimensional ray space, including subpixel and lens coordinates, allows our (c) Area ReSTIR to reuse even more paths (despite bokeh and high-frequency content) also while shading just one sample per pixel. This approaches the quality of (d) converged path tracing. Note: (a), (b), and (c) all use the same number of independent paths each pixel.*

Recent advancements in spatiotemporal reservoir resampling (ReSTIR) leverage sample reuse from neighbors to efficiently evaluate the path integral. Like rasterization, ReSTIR methods implicitly assume a pinhole camera and evaluate the light arriving at a pixel through a single predetermined subpixel location at a time (e.g., the pixel center). This prevents efficient path reuse in and near pixels with high-frequency details.

We introduce *Area ReSTIR*, extending ReSTIR reservoirs to also integrate each pixel's 4D ray space, including 2D areas on the film and lens. We design novel subpixel-tracking temporal reuse and shift mappings that maximize resampling quality in such regions. This robustifies ReSTIR against high-frequency content, letting us importance sample subpixel and lens coordinates and efficiently render antialiasing and depth of field.

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

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

*Joint first authors; equal contribution.

Authors' addresses: Song Zhang, song.zhang@utah.edu, University of Utah, USA; Daqi Lin, daqil@nvidia.com, NVIDIA, USA; Markus Kettunen, mkettunen@nvidia.com, NVIDIA, Finland; Cem Yuksel, cem@cemyuksel.com, University of Utah, USA; Chris Wyman, chris.wyman@acm.org, NVIDIA, USA.

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2024 Copyright held by the owner/author(s).
0730-0301/2024/7-ART98
https://doi.org/10.1145/3658210

ACM Reference Format:
Song Zhang, Daqi Lin, Markus Kettunen, Cem Yuksel, and Chris Wyman. 2024. Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing. *ACM Trans. Graph.* 43, 4, Article 98 (July 2024), 13 pages. https://doi.org/10.1145/3658210

1 INTRODUCTION

Reservoir-based spatiotemporal importance resampling, or ReSTIR [Bitterli et al. 2020], uses weighted reservoir sampling [Chao 1982] to aggregate spatial and temporal neighbor samples in a streaming fashion, giving effective sample counts up to 100× higher than naive independent sampling. This has been applied to various rendering problems: many-light sampling [Bitterli et al. 2020], diffuse global illumination [Ouyang et al. 2021], and more general light paths [Lin et al. 2022] including volumetric scattering [Lin et al. 2021].

But these algorithms shade only at pixel centers, essentially using a delta function as the pixel filter; this introduces aliasing. But more interesting pixel filters cause problems. Near subpixel detail, such as foliage or hair, randomizing subpixel locations forcibly changes the reused path's primary hit, making it difficult to reuse secondary path vertices from neighbors. Thus, high-frequency detail often causes significant quality degradation.

We introduce *Area ReSTIR,* which expands ReSTIR to sample pixel footprints. Our samples estimate radiance over the entire pixel rather than a single point. Besides pixel areas, our method also integrates over camera aperture, resampling primary rays in a 4D ray space. The intuition goes: ReSTIR PT [Lin et al. 2022] already estimates

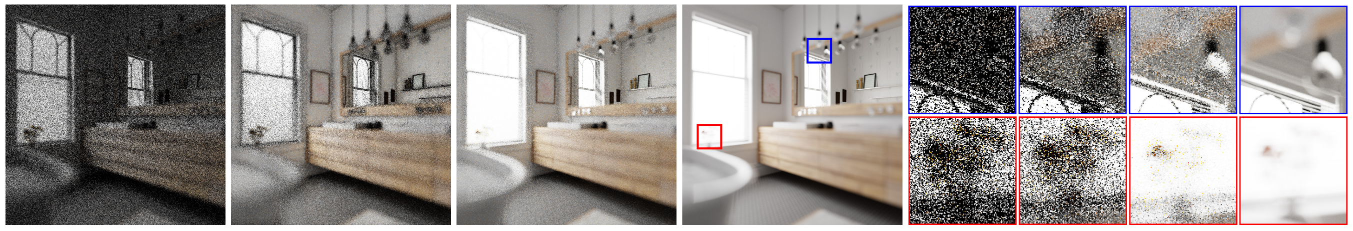

(a) Path tracing, 4.4 ms
(b) ReSTIR PT, 13.8 ms
(c) Area ReSTIR, 16.0 ms
(d) Ground Truth
(a)
(b)
(c)
(d)

Fig. 2. *Depth of field in the* Contemporary Bathroom *with focal plane on the (reflected) far wall. (a) With one independent path per pixel, no reuse occurs. (b) ReSTIR PT [Lin et al. 2022] reuses samples, but struggles to identify reusable paths in defocused areas with complex geometry, specular effects, or edge discontinuities. (c) Our new Area ReSTIR reuses significantly more samples in these regions, as we analyze the signal in the measurement equation's higher-dimensional space. (d) A converged path traced reference.*

high-dimensional integrals over complex domains; we should be able to include a few more dimensions, e.g., subpixel location (u, v) and aperture (s, t).

Instead of reusing samples to solve the rendering equation for one subpixel location at a time, we reuse to solve the higher-dimensional *measurement equation*, which integrates over the pixel filter and camera parameters, enhancing analysis of overlaps among adjacent pixel domains or reservoirs. Counterintuitively, estimating a higher-dimensional integral increases quality at a given sampling rate. By enabling more effective reuse, fewer pixels discard their history and must restart aggregation; this reduces noise (e.g., see Figure 1).

We initially aimed to reduce aliasing with ReSTIR, but discovered our method also improves reuse on areas with e.g., high-frequency normal maps and geometry. Our Area ReSTIR also naturally extended to depth of field. This follows the general direction of extending reuse dimensions in ReSTIR literature (see Figure 2); ReSTIR was first designed to reuse direct illumination, then extended to global illumination, and now also to subpixel and aperture coordinates.

Our specific contributions include:

- Importance sampling pixel footprint and depth of field by applying ReSTIR to the measurement equation,
- Subpixel-tracking temporal reuse by reprojection with *non-integer* motion vectors, drawing samples from multiple overlapping prior reservoirs (Section 4.2),
- Robustifying temporal reuse in pixels with high-frequency geometry or normals (Section 4.3),
- A new shift mapping and MIS weights to improve sample reuse for depth of field (Sections 4.6 and 5.3),
- Direct support for advanced (non-box) pixel filters in ReSTIR, which may improve temporal reuse (Section 4.4).

We show that when shading just one sample per pixel, spatiotemporal sample reuse allows complex primary ray effects in real-time, including antialiasing and depth of field.

2 RELATED WORK

Renderers typically solve the intensity of a pixel by the *measurement equation* [Veach 1998], integrating radiance over all paths through a pixel. Most algorithms do this by point sampling the *rendering equation* [Kajiya 1986] for one or more incident directions. Recovering the accurate intensity from few point samples spawned entire new research fields (e.g., postprocess antialiasing [Yang et al. 2020] and depth of field [Demers 2004]), to perform higher-quality filtering despite a low sampling budget.

2.1 Spatiotemporal Reservoir Resampling

We extend reservoir-based spatiotemporal importance resampling (ReSTIR) [Bitterli et al. 2020] to solve the measurement equation by explicitly integrating over camera and pixel footprints. Generally, ReSTIR enables complex sample reuse, even between varied integration domains, by unbiased resampling and reweighting using reservoir sampling [Chao 1982] to efficiently leverage modern GPUs' stream computation model.

Bitterli et al. [2020] applied ReSTIR to direct lighting, and it has been extended to diffuse global illumination [Jiang et al. 2023; Ouyang et al. 2021], and more complex paths [Lin et al. 2022, 2021]. Recently, Chang et al. [2023] and Wang et al. [2023] both adapt ReSTIR to differentiable rendering, and Sawhney et al. [2024] integrates Markov Chain Monte Carlo mutations with ReSTIR to reduce correlations. But most of these methods use reservoirs to estimate radiance through a single point in screen space and reuse only subpaths leaving the corresponding primary hit point.

A few techniques apply ReSTIR in other spaces. Kettunen et al. [2023] resample in conditional probability spaces, Weinrauch et al. [2023] use reservoirs in object space, and Boissé [2021] stores reservoirs in world space hash grids. Boksansky et al. [2021] use reservoirs in voxels, but keeping this unbiased is tricky. Pixels using voxels with high-frequency geometry may see arbitrarily different parts of the scene (compared to voxel centers); this is hard to track. And as Wyman et al. [2023] emphasize, knowing and accounting for supports is key to keeping ReSTIR unbiased.

Area ReSTIR breaks the convention of integrating at points. Our reservoirs act as estimators over an integration domain *including* pixel footprint and camera aperture. A reservoir is no longer associated with a single point, but a region. If we jitter the pixel (u, v) or aperture (s, t), we can often continue reusing our temporal history. These area sample domains overlap, so many samples can still help estimate a future frame's (higher-dimensional) integral.

2.2 Gradient-Domain Rendering

Researchers have explored other path reuse techniques. Bekaert et al. [2002] reuse samples within a single integration domain and Bauszat et al. [2017] expand to reuse between domains with appropriate *shift mappings*. Shift maps come from gradient-domain rendering [Kettunen et al. 2015; Lehtinen et al. 2013], which computes discrete image gradients between a pixel sample and shifted neighbor.

Many shift mappings exist (see Hua et al.'s [2019] survey), but Lin et al. [2022] design a *hybrid shift*, combining random replay and reconnection shifts, optimized for ReSTIR-based path reuse on the GPU. In all previous shift mappings, primary hits are given as fixed, and the shift mapping produces the rest of the path vertices.

In Area ReSTIR, we must also decide how to shift map primary hits (i.e., subpixel locations) between only partially overlapping pixel footprints (due to motion between frames). In this paper, we extend existing shift mappings to this context and define a new shift mapping to improve bokeh of directly visible emissives.

2.3 Depth of Field

Frequently, renderers approximate depth of field from a thin-lens camera using a postprocess blur [Potmesil and Chakravarty 1982]. Distributed ray tracing [Cook et al. 1984] and accumulation buffers [Haeberli and Akeley 1990] use Monte Carlo sampling to accurately estimate depth of field, but both are generally infeasible in real time.

Specialized reconstruction filters improve the quality of such Monte Carlo estimates [Bako et al. 2017; Belcour et al. 2013; Li et al. 2012], but these typically require providing relatively high sample counts as input, limiting use in real time.

Real-time applications typically postprocess a pinhole camera image, often split into multiple layers using depth peeling [Everitt 2001], and then processing with pyramidal processing [Kraus and Strengert 2007], diffusion [Kass et al. 2006], or layer traversal via ray queries [Lee et al. 2009]. Production games often use a complex mix of multiple techniques [Abadie 2018; Jimenez 2014] to ameliorate the artifacts inherent in postprocessing, including light leaking, over-blurring, ghosting, and hallucination (of occluded samples).

Our work instead expands integration to include camera aperture, letting ReSTIR integrate depth of field. Shading only a single sample per pixel, we get an unbiased, high-quality estimate.

2.4 Antialiasing

Renderers sampling pixels at few deterministic locations (e.g., pixel centers) leave the image aliased, and require postprocess filtering to remove these artifacts. This has challenged real-time renderers for decades, leading to techniques like hardware multisampling [Akeley 1993], inferring edges from the image [Jimenez et al. 2011; Lottes 2009; Reshetov 2009], or the recently ubiquitous temporal antialiasing [Karis 2014; Korein and Badler 1983; Yang et al. 2020].

More recently, researchers and developers have leveraged neural networks to antialias and upsample images (e.g., DLSS and XeSS) [Hasselgren et al. 2020; NVIDIA 2019; Xiao et al. 2020]. After long training over giant datasets, these networks predict high-quality weights for nearby spatiotemporal pixels to guess an antialiased image from aliased inputs (plus some other benefits, like denoising).

Compared to point-sampling real-time renderers, our work expands the integration domain over pixel footprints, allowing us to deterministically apply MIS weighting [Veach and Guibas 1995] to spatiotemporally reuse samples without any training process. This spatiotemporal reuse gives an unbiased estimate of the perfectly antialiased image, while only shading one sample per pixel.

3 PRELIMINARIES

Before introducing our work, we first review ReSTIR and key related concepts. See Wyman et al. [2023] for a more in-depth review.

3.1 Resampling and ReSTIR

Resampled importance sampling (RIS) [Talbot et al. 2005] provides an unbiased way to estimate an integral $\int_\Omega f(x)\,\mathrm{d}x$ using samples Y *resampled* from a set of M independent candidates X_j. By drawing candidates with a cheap distribution $p(x)$ and resampling with user-selected target function $\hat{p}(x)$, Y's sample distribution converges to $\bar{p}(x) = \hat{p}(x)/\int_\Omega \hat{p}(x)\,\mathrm{d}x$ as M grows. Picking $\hat{p} \approx f$ means such Y estimates the integral of f with low variance. This has the form

$$\langle I \rangle_{\mathrm{RIS}} = f(Y)W_Y, \quad \text{where } W_Y = \frac{1}{\hat{p}(Y)} \sum_{j=1}^{M} \frac{1}{M} \frac{\hat{p}(X_j)}{p(X_j)}, \qquad (1)$$

where the sum better approximates $\int_\Omega \hat{p}(x)\,\mathrm{d}x$ as M increases, i.e., $W_Y \approx 1/\bar{p}(Y)$.

Generalized RIS (GRIS) [Lin et al. 2022] formally defines an *unbiased contribution weight* (UCW) W_X as any weight satisfying $\mathbb{E}[W_X|X] = 1/p_X(X)$, meaning Monte Carlo integration can use these weights instead of a sample PDF (as in Equation 1, left).

Lin et al. [2022] show UCWs can replace not just simple PDFs; UCWs often exist where exact PDFs are impossible to practically evaluate, as in resampling over frames with different per-pixel integration domains. If reusing from varied domains, a per-domain shift mapping T_j maps candidates X_j into $T_j(X_j)$ in the common target domain. In this case, a more general form of Equation 1 is

$$W_Y = \frac{1}{\hat{p}(Y)} \sum_{j=1}^{M} w_j, \quad \text{for } w_j = m_j\big(T_j(X_j)\big)\, \hat{p}\big(T_j(X_j)\big)\, W_{X_j} \left|\frac{\partial T_j}{\partial X_j}\right|, \quad (2)$$

where w_j are per-candidate *resampling weights*, m_j are *resampling MIS weights*, and Y is chosen from $T_j(X_j)$ proportionally to w_j. The candidates are treated as different sampling strategies, and these MIS weights normalize their coverage over the target domain.

Mathematically, the m_j form a partition of unity over these strategies, so every target domain point y has $\sum_{j=1}^{M} m_j(y) = 1$. If strategy j cannot generate y with positive PDF, then $m_j(y) = 0$. Incorrect m_j, as when assuming constant $m_j = 1/M$ during spatiotemporal reuse, are a common cause of bias when implementing ReSTIR.

When reusing from many domains, the union of candidate supports must cover the target domain (any path with $\hat{p} > 0$ must be producible by at least one strategy). To guarantee coverage, we take at least one *canonical sample* (a sample from the current pixel). This is one role played by new independent, per-pixel candidates in each frame.

GRIS allows resampling passes to be iteratively chained without bias. This enables the feed-forward chain of sample distributions, improving over time, that characterizes ReSTIR [Bitterli et al. 2020].

3.2 Efficient Streaming Sampling

Efficient feed-forward reuse relies on *weighted reservoir sampling* [Chao 1982], a constant-space algorithm to randomly pick one element from a stream. It has many nice properties, including the ability to combine choices from many streams without replaying any of them. This allows paths to improve render quality long after they no longer reside in memory, as only chosen paths must be re-evaluated each frame. Other samples contribute indirectly via the chosen sample's weight and distribution.

Weighted reservoir sampling uses a *reservoir* data structure to store its current state. Fundamentally, reservoirs only need to store the current selected sample Y and a running sum of weights (equivalently, either W_Y or $\sum_{j=1}^{M} w_j$). But ReSTIR often localizes reservoirs to a particular sampling location, so reservoirs may also include, e.g., a pixel ID, path or shading data, a count of seen samples, or other implementation dependent details. After sampling a new path, the reservoir is updated with a short, simple algorithm (see Bitterli et al. [2020], Algorithm 2).

3.3 Issues with the Measurement Integral

Path tracing computes radiance by solving the rendering equation [Kajiya 1986],

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_{S^2} \rho(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L(\mathbf{x}, \boldsymbol{\omega}_i) \, |n(\mathbf{x}) \cdot \boldsymbol{\omega}_i| \, \mathrm{d}\boldsymbol{\omega}_i, \quad (3)$$

with incident direction $\boldsymbol{\omega}$ integrated over the sphere S^2, and $L(\mathbf{x}, \boldsymbol{\omega}_o)$, the outgoing radiance in direction $\boldsymbol{\omega}_o$ from surface point $\mathbf{x}$ with normal $n(\mathbf{x})$. This surface may emit light L_e, or reflect other incident light L from directions $\boldsymbol{\omega}_i \in S^2$ according to the surface BSDF ρ.

The intensity of pixel j is determined by the measurement equation [Veach and Guibas 1997],

$$I_j = \int_{\Omega} h_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}, \quad (4)$$

given as an integral over $\bar{x} \in \Omega$, all paths connecting emitters to the sensor, and h_j is the pixel filter, f is the measurement contribution function. We reparametrize the primary hit $\mathbf{x}_1$ with image plane position $\mathbf{u} \in \mathcal{U} = [0, H] \times [0, W]$. Separating image coordinates from the rest of the path $\hat{\mathbf{x}} = \mathbf{x}_2 \mathbf{x}_3 \cdots \in \hat{\Omega}$ gives:

$$I_j = \int_{\mathcal{U}} \left(\int_{\hat{\Omega}} h_j(\mathbf{u}) f(\mathbf{u}, \hat{\mathbf{x}}) \, \mathrm{d}\hat{\mathbf{x}} \right) \mathrm{d}\mathbf{u}, \quad (5)$$

where $h_j(\mathbf{u})$ is the pixel filter with $\int_{\mathcal{U}} h_j(\mathbf{u}) \, \mathrm{d}\mathbf{u} = 1$.

Prior ReSTIR work estimates I_j by fixing image coordinates $\mathbf{u}_j \in \mathbb{R}^2$ inside pixel j each frame, spatiotemporally sharing subpaths in $\hat{\Omega}$ to integrate

$$\hat{I}_j(\mathbf{u}_j) = \int_{\hat{\Omega}} h_j(\mathbf{u}_j) f(\mathbf{u}_j, \hat{\mathbf{x}}) \, \mathrm{d}\hat{\mathbf{x}} \quad (6)$$

for each pixel. Earlier work assumes a box filter, supporting other kernels by splatting [Lin et al. 2022].

For best results, ReSTIR should choose target functions matching the integrand, which in prior work means $\hat{p}(\hat{\mathbf{x}}) = f(\mathbf{u}_j, \hat{\mathbf{x}})$, as $\mathbf{u}_j$ is within pixel j. Randomizing $\mathbf{u}_j$ mutates ReSTIR's target distribution, potentially invalidating temporal reuse where f changes quickly due to high-frequency normal maps, glinty materials, hair, or foliage. Camera or object motion can degrade reuse quality in these regions even if using fixed $\mathbf{u}_j$ at pixel centers, but the issues are magnified when rendering with camera jittering for antialiasing.

4 AREA RESTIR

Our Area ReSTIR improves robustness near high frequencies occurring within pixels by targeting the full path space $\mathcal{U} \times \hat{\Omega}$ to integrate I_j instead of the varying $\hat{I}_j(\mathbf{u}_j)$. In other words, we let ReSTIR *also* importance sample subpixel locations. We store samples over pixel footprints in *area reservoirs*, defined in Section 4.1. Area reservoirs no longer always reside on a pixel grid, so we analyze how fractional offsets affect reuse in Section 4.2.

We extend Area ReSTIR to also improve depth of field by letting ReSTIR control the coordinates $\mathbf{s} \in \mathcal{S} = [0, 1]^2$ that map to lens area in a finite-aperture camera, e.g. via disk sampling. Pixel intensity is then given by

$$I_j = \int_{\mathcal{U} \times \mathcal{S} \times \hat{\Omega}} h_j(\mathbf{u}) f(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}) \, \mathrm{d}(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}). \quad (7)$$

The ideal target function[1] becomes $\hat{p}(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}) = h_j(\mathbf{u}) f(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$, and Area ReSTIR spatiotemporally reuses triplets $(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ via shift mappings.

Prior methods reprojected, shifted, and reused samples on a pixel grid at integer precision. To achieve good subpixel-accurate reuse, we must carefully design our new temporal reuse (see Section 4.3) as multiple prior-frame reservoirs may now contain reusable samples (see Figure 3); we discuss spatial reuse in Section 4.5.

Earlier work [Lin et al. 2022] only shifts $\hat{\mathbf{x}}$ and randomizes earlier path vertices. As those randomized vertices are vital for good path reuse with depth of field, we design a new shift mapping specifically to handle defocused areas (see Section 4.6).

Sample reuse with area reservoirs, target function $\hat{p}(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$, our temporal reuse, and enhanced shift mapping greatly improve reuse robustness compared to prior work (see Figure 1). While we start by assuming area integration with a box filter, our method also supports more advanced filters h_j in $\hat{p}$ (see Section 4.4).

4.1 Area Reservoirs

As our reservoirs now span a screen-space area and the path space behind it ($\mathrm{supp}(h_j) \times \hat{\Omega}$), we call these *area reservoirs* to distinguish them from prior work's *point reservoirs* covering path space behind a *single point* (i.e., $\{\mathbf{u}_j\} \times \hat{\Omega}$). Our h_j may be any common filter with finite support around pixel centers; we start by assuming a box filter before later generalizing.

Area reservoirs store paths parametrized as $\boldsymbol{x} = (\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ along with their unbiased contribution weights $W_{\boldsymbol{x}}$. Image-space location $\mathbf{u}$ lies in h_j's support and lens coordinate $\mathbf{s}$ lies in $[0, 1)^2$. Here, $\mathbf{u}$ and $\mathbf{s}$ define path vertices $\mathbf{x}_0$ and $\mathbf{x}_1$, and $\hat{\mathbf{x}}$ defines subsequent vertices. We use path contribution $\hat{p}_j(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}) = h_j(\mathbf{u}) f(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ as our target function. A key takeaway: area reservoirs need not have integer $\mathbf{u}$ or be fixed to pixel centers, but as in Figure 3, reservoir footprints may be offset from the pixel grid.

To get samples, we trace candidates by generating path $\boldsymbol{x}$. We sample $\mathbf{u}$ proportionally to the pixel filter, $p_{\mathbf{u}} = h_j$, and $\mathbf{s}$ uniformly

[1] $|h_j(\mathbf{u})|$ should be used in $\hat{p}$ if the filter (e.g. Mitchell–Netravali [1988]) has negative values (as $\hat{p}$ cannot be negative). We drop the absolute value for simplicity.

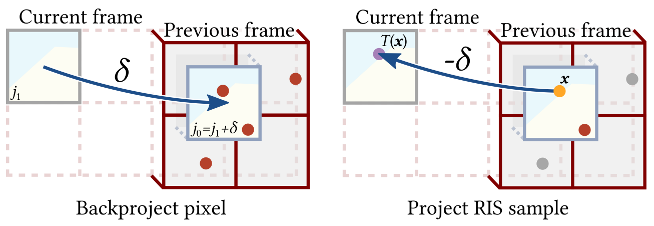


Fig. 3. *Our subpixel-aware shift map first backprojects the current pixel to the previous frame by motion vector δ (left). The projected pixel's support overlaps 2×2 prior pixels, each with one reservoir sample (red). We select one via RIS, discarding those outside the support (gray). We then reproject the chosen sample (orange) forward to the current frame by shifting by $-\delta$.*

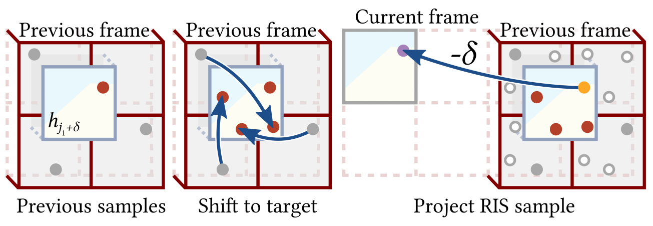


Fig. 4. *Our advanced shift reduces variance by always giving RIS the same number of candidates (right). Prior frame samples outside the pixel support (gray) are shifted to the target pixel but retain their subpixel coordinates (middle). We select one by RIS (orange) and reproject it to the current frame by a shift of $-\delta$. Conceptually, we shift all prior frame samples to their 2×2 neighbors, but these virtual samples (white) are only needed for MIS weights and proving correctness.*

from $[0, 1)^2$. These define path vertices $\mathbf{x}_0$ and $\mathbf{x}_1$, and remaining vertices $\hat{\mathbf{x}}$ are produced by path tracing. This gives initial path candidates a PDF $p(\boldsymbol{x}) = 1 \cdot p_{\mathbf{u}}(\mathbf{u}) p(\hat{\mathbf{x}}|\mathbf{u}, \mathbf{s})$. While we initially sample subpixel location $\mathbf{u}$ proportional to the pixel filter, reuse via ReSTIR quickly improves this subpixel distribution.

4.2 Fractional Matching in Area Reservoirs

Previous resampling methods used *integer backprojection*, i.e., projecting to the nearest discrete prior frame pixel. This is problematic where small subpixel offsets drastically impact the path contribution. Area reservoirs improve robustness due to *fractional backprojection*, finding reservoirs with (partially) overlapping integration domains, resampling with appropriate MIS weights, and allowing subpixel-precise reuse.

Let us compare a simplified version of our subpixel-aware reuse to the older method. Consider a static scene with moving camera. Assume the camera moves from frame 0 to 1 such that pixel j_1 exactly corresponds to off-grid pixel[2] $j_0 = j_1 + \delta$ in the previous frame, where δ may be fractional. Any $\mathbf{x} = (\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ in j_1 in frame 1 and $(\mathbf{u} + \delta, \mathbf{s}, \hat{\mathbf{x}})$ in j_0 in frame 0 represent the same geometric path, yielding the same path contribution in the two frames, i.e. $f_1(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}) = f_0(\mathbf{u} + \delta, \mathbf{s}, \hat{\mathbf{x}})$.

Following Figure 3, we can define a temporal shift from frame 0 to 1, $T(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}}) = (\mathbf{u} - \delta, \mathbf{s}, \hat{\mathbf{x}})$ that translates $\mathbf{u}$ by the motion vector and keeps $\mathbf{s}$ and $\hat{\mathbf{x}}$ the same. This yields the same geometric path.

Sample reuse quality largely depends on how well the two frames' target functions match, with ratios close to one lowering resampling variance [Wyman et al. 2023]. Assuming a box filter and denoting the target function of frame i by $\hat{p}_i$, we get ratio

$$\frac{\hat{p}_1(T(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0))}{\hat{p}_0(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0)} = \frac{f_1(\mathbf{u}_0 - \delta, \mathbf{s}_0, \hat{\mathbf{x}}_0)}{f_0(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0)} = \frac{f_0((\mathbf{u}_0 - \delta) + \delta, \mathbf{s}_0, \hat{\mathbf{x}}_0)}{f_0(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0)} = 1. \quad (8)$$

Consider the point reservoirs in the same situation. We read $\hat{\mathbf{x}}_0$ from reservoir $j + \text{round}(\delta)$. The target function ratio is

$$\frac{\hat{p}_1(T(\hat{\mathbf{x}}_0))}{\hat{p}_0(\hat{\mathbf{x}}_0)} = \frac{\hat{p}_1(\hat{\mathbf{x}}_0)}{\hat{p}_0(\hat{\mathbf{x}}_0)} = \frac{f_1(\mathbf{u}_1, \mathbf{s}_1, \hat{\mathbf{x}}_0)}{f_0(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0)} = \frac{f_0(\mathbf{u}_1 + \delta, \mathbf{s}_1, \hat{\mathbf{x}}_0)}{f_0(\mathbf{u}_0, \mathbf{s}_0, \hat{\mathbf{x}}_0)}, \quad (9)$$

where $\mathbf{u}_0, \mathbf{u}_1, \mathbf{s}_0$ and $\mathbf{s}_1$ are random, and $\mathbf{u}_1 + \delta$ and $\mathbf{u}_0$ may be different. If f_0 varies fast as other parameters change, the ratio may vary wildly. But if f_0 is insensitive to other parameters, reuse may be efficient.

[2]We explain how to populate this off-grid pixel with reusable samples in Section 4.3.

This also explains why prior ReSTIRs are sensitive to visual frequencies, even without jittering or depth of field. Denote $d = \text{round}(\delta)$ and assume shading at pixel centers. Then,

$$\frac{\hat{p}_1(T(\hat{\mathbf{x}}_0))}{\hat{p}_0(\hat{\mathbf{x}}_0)} = \frac{\hat{p}_1(\hat{\mathbf{x}}_0)}{\hat{p}_0(\hat{\mathbf{x}}_0)} = \frac{f_0(j_1 + \delta + 0.5, \hat{\mathbf{x}}_0)}{f_0(j_1 + d + 0.5, \hat{\mathbf{x}}_0)}, \quad (10)$$

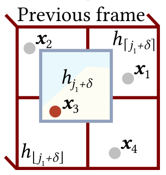
i.e., subpixel detail may degrade reuse for any non-integer motion.

4.3 Temporal Reuse

Area ReSTIR reduces this frequency sensitivity by carefully matching paths at subpixel precision for temporal reuse. To temporally reuse area reservoirs, we backproject current filter h_{j_1}'s footprint into the previous frame. See Figure 3, which depicts a box filter; Section 4.4 generalizes to finite support filters. Backprojection can map to an arbitrary pixel-sized area. We use a real-time convention, approximating motion with a translation along a motion vector δ, leaving more general projections as future work. We then select an overlapping sample from the prior frame by RIS, and finish by shifting the selection back $-\delta$ to the current frame.

We outline two options for identifying previous frame samples: a fast option that picks from samples overlapping the backprojected pixel (Figure 3), and a more expensive and robust option that ensures all nearby samples can contribute, by shifting them onto the pixel's support (Figure 4). We start with the fast option, which is simpler to implement and provides a constructive step towards realizing the robust version. Additionally, it allows trading some quality to achieve closer to baseline performance.

4.3.1 *Option (Fast): Using samples as-is.*



Consider resampling path $\boldsymbol{x}$ from four inputs $\boldsymbol{x}_k = (\mathbf{u}_k, \hat{\mathbf{x}}_k)$ from area reservoirs overlapping the backprojected pixel's footprint $h_{j_1+\delta}$ in the prior frame. If pixel coordinate $\mathbf{u}_k$ lies in kernel $h_{j_1+\delta}$, sample $\boldsymbol{x}_k$ gets a positive resampling weight with $\hat{p}_{j_1+\delta}(\boldsymbol{x}_k) = h_{j_1+\delta}(\mathbf{u}_k) f(\mathbf{u}_k, \hat{\mathbf{x}}_k)$; we reuse with the identity shift. Otherwise we consider $\boldsymbol{x}_k$ invalid, with no contribution. This gives MIS weights

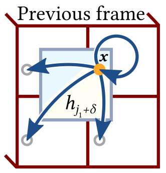
$$m_k(y) = \mathbb{1}_{\text{supp}\, \boldsymbol{x}_k}(y) = \begin{cases} 1 & y \in \text{supp}\, \boldsymbol{x}_k \\ 0 & \text{otherwise} \end{cases}. \quad (11)$$

This trivially satisfies $\sum_{k=1}^{4} m_k(y) = 1$, guaranteeing unbiased integration (over supp $h_{j_1+\delta}$) using GRIS (see Section 2.1).

While the fast option is unbiased and reduces variance from high frequency content, it is imperfect: prior frame samples can only be reused if they lie in backprojected filter $h_{j_1+\delta}$. Such samples may not exist; $\boldsymbol{x}_k$ may all lie outside of the reprojected pixel's area ($h_{j_1+\delta}$), in which case temporal reuse returns a null sample. This randomly resets temporal history in motion, and is thus not an ideal reuse solution in all cases. We still report results with this fast option due to its good performance, but we default to use the following, more robust option.

4.3.2 Option (Robust): Guaranteeing coverage. To address this problem, we propose adjusting the positions of samples $\boldsymbol{x}_k$ located outside of the filter $h_{j_1+\delta}$ by one-pixel shifts. This brings outside samples back to the filter range (see Figure 4).

For 2×2 prior frame reservoirs, imagine mapping each sample $\boldsymbol{x}_k$ to all neighbors via one-pixel shifts vertically, horizontally and diagonally (Figure 4, center). Exactly one (potentially shifted) copy of each will lie in backprojected pixel $h_{j_1+\delta}$; the other shifts need not be executed (Figure 4, white circles). This allows reusing all four neighbor samples, ideally as-is, but shifting into $h_{j_1+\delta}$ as a backup.



For MIS weights to combine these four candidates, we use the generalized balance heuristic [Lin et al. 2022]. This requires knowing all neighbors' probability densities of generating the selected sample $\boldsymbol{x} = (\mathbf{u}, \hat{\mathbf{x}})$, with $\mathbf{u}$ in $h_{j_1+\delta}$. We shift $\boldsymbol{x}$ to all four pixels and evaluate the pixels' target functions as proxies for the unknown PDFs in the balance heuristic. One of the shifts is identity and returns $\boldsymbol{x}$.

Considering quality, this method improves on the option in Section 4.3.1. When backprojected filter $h_{j_1+\delta}$ overlaps prior samples, they provide good low-variance reuse. The balance heuristic bounds variance degradation [Veach 1998] from the worst neighbors. When all samples are outside $h_{j_1+\delta}$'s support, the fast option produces a null sample. The robust option can avoid resetting temporal history by borrowing neighbor samples; if just one shift succeeds, our current-frame pixel inherits that pixel's history.

4.3.3 Mapping Back to the Current Frame. After choosing a prior frame sample to reuse (from $h_{j_1+\delta}$), we shift this sample back to the current frame by subtracting the motion vector δ from its $\mathbf{u}$ coordinate, and apply GRIS to combine with the canonical sample from current frame's pixel. (This canonical sample is the new independent path shot each frame in every pixel.)

Variance depends mainly on how well area reservoirs overlap between frames, errors due to approximating backprojection via translation, and how well the used shift mapping adapts to temporal changes to the scene.

4.4 Extension to General Pixel Filters

For general pixel filters, we assume each filter h_j's support is a rectangle containing pixel j [Pharr et al. 2016]. The analysis in Sections 4.2 and 4.3 largely still applies; we still backproject filter h_j to the prior frame as $h_{j+\delta}$, overlapping prior area reservoirs. But prior reservoir contributions are no longer disjoint, which requires reworking the MIS weights in Equation 11 to account for overlaps:

$$m_k(y) = \frac{\beta_k h_k(y)}{\sum_{k'=1}^{4} \beta_{k'} h_{k'}(y)}, \qquad (12)$$

where β_k are bilinear interpolation weights for the backprojected filter relative to the four overlapping reservoirs.

This accounts for filter importance and increases the weight of better-matching areas: if the backprojected filter exactly aligns with a single temporal reservoir, that will be the only reservoir used. Box filters (Equation 11) are a special case of Equation 12.

Theoretically, with larger pixel filter footprints, samples could be borrowed from a larger number of pixels, but this is not required for unbiased results. We leave optimizing this for future work.

4.5 Spatial Reuse

Fractional motion vectors help our temporal sample reuse. But we do not use fractional motion vectors for spatial reuse: that requires first finding subpixel correspondences between nearby pixels.

Instead, like prior work [Bitterli et al. 2020; Lin et al. 2022, 2021], we sample random neighbors from a screen-space neighborhood at integer distances (essentially, integer δ) and rely on shift mappings between their domains and careful MIS weighting for spatial reuse.

Moving to offline rendering, where larger pixel filters may be desirable, the integration domains of very nearby spatial neighbors may once again overlap, making the shift in Section 4.3.1 also interesting for spatial reuse.

However, unlike prior work, our spatial shift maps can also map lens coordinates (as described in Section 4.6). This improves spatial reuse in the presence of depth of field.

4.6 Depth of Field

When reusing sample $\boldsymbol{x} = (\mathbf{u}, \hat{\mathbf{x}})$ in Section 4.3, we left the shift map unspecified, e.g., $\hat{\mathbf{x}}$ could be shifted by vertex copy ($T(\hat{\mathbf{x}}) = \hat{\mathbf{x}}$) [Lehtinen et al. 2013] (e.g. when building Area ReSTIR on ReSTIR DI) or by Lin et al.'s [2022] hybrid shift (when building on ReSTIR PT).

For non-pinhole cameras, we must also shift lens position $\mathbf{s}$ or, equivalently, path vertex $\mathbf{x}_0$. Generalizing, our *lens vertex copy* reuses lens coordinate $\mathbf{s}$, extending how prior methods reuse pinhole camera paths (where $\mathbf{x}_0$ must remain fixed). This shift should work for small apertures and has a Jacobian of 1.

But large apertures can have big circles of confusion. This spreads bright lights (or reflections) at primary hitpoints $\mathbf{x}_1$ into large, glowing *bokehs* with shape dependent on camera aperture. Since $\mathbf{x}_0$ and $\mathbf{x}_1$ together define the pixel, a shift mapping that reuses the camera vertex $\mathbf{x}_0$ *necessarily* must have different primary hit $\mathbf{x}_1$ for neighbor pixels. Thus, prior shifts fail to significantly reduce variance inside bokehs, as $\mathbf{x}_1$ is not reused.

We introduce a new shift mapping that reuses vertex $\mathbf{x}_1$. Instead, we produce the shifted camera position $\mathbf{x}'_0$ by projecting $\mathbf{x}_1$ through the focal plane at sample $\mathbf{u}'$ onto the lens and compute $\mathbf{s}'$ corresponding to $\mathbf{x}'_0$ (see Figure 5). We call this the *primary hit reconnection* shift and derive its Jacobian in Appendix A.

As with any shift map, this is not universally applicable. Reprojecting $\mathbf{x}_1$ onto the lens may hit a point $\mathbf{x}'_0$ outside the aperture. The

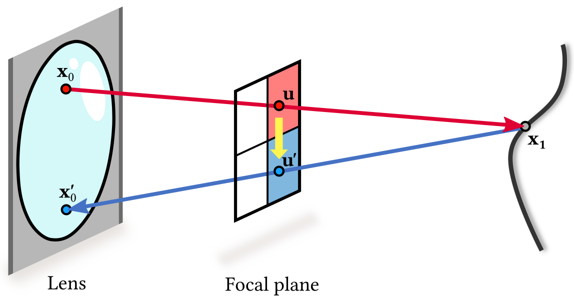


Fig. 5. *Computing new lens position* $\mathbf{x}_0'$ *using the* primary hit reconnection shift. *Project the input path's primary hit* $\mathbf{x}_1$ *back to the camera through the shifted path's position* $\mathbf{u}'$ *on the focal plane and compute* $\mathbf{s}'$ *corresponding to* $\mathbf{x}_0'$*. The shift fails if* $\mathbf{x}_0'$ *is outside the lens or the ray* $\mathbf{x}_1 \to \mathbf{x}_0'$ *is occluded.*

```
struct pointReservoir {
    Sample x;  // Light or path sample
    float  W;  // Unbiased contribution weight
    uint   M;  // Sample count included
//  Implicit:     Sample position (0.5, 0.5)
//  Implicit:     Lens position (at pinhole)
//  Implicit:     Pixel ID (in [0,W) x [0,H))
};
```

```
struct areaReservoir {
    Sample x;  // Light or path sample
    float  W;  // Unbiased contribution weight
    uint   M;  // Sample count included
    float2 u;  // Sample pos relative to reservoir corner
    float2 s;  // Lens position (in [0,1) x [0,1))
//  Implicit:     Pixel ID (in [0,W) x [0,H))
};
```

Fig. 6. *Key differences between prior ReSTIR reservoirs (e.g., Bitterli et al. [2020]) and our area reservoirs. Previously, subpixel sample and camera locations were fixed per frame, allowing reservoirs to implicitly reference them. Area reservoirs must store a lens sample and the fractional subpixel location, relative to the reservoir boundary.*

primary ray between $\mathbf{x}_1$ and reprojected $\mathbf{x}_0'$ could also be occluded. In either case, the shift becomes invalid.

For small apertures and vertices $\mathbf{x}_1$ near the focal plane, primary hit reconnections are unlikely to succeed. Using MIS between our two shifts thus often produces a valid sample. Generalized RIS [Lin et al. 2022] can handle this MIS by duplicating a reservoir and assigning different shifts to each. Alternatively, half of our randomly selected spatial neighbors can be assigned each shift; this typically improves performance significantly.

5 IMPLEMENTATION DETAILS

Area ReSTIR can augment any screen-space ReSTIR algorithm. We built upon Bitterli et al.'s [2020] direct illumination algorithm for simplicity, while also adapting ReSTIR PT [Lin et al. 2022] to verify the generalizability of our approach. Area ReSTIR affects just lens and primary hit points, so this was straightforward.

5.1 Reservoirs, Reuse, and G-Buffers

Extending to area reservoirs is fairly simple, as shown in Figure 6. Previously implicit data like lens sample $\mathbf{s}$ and fractional subpixel positions $\mathbf{u}$ must be stored. Storing these values in the range $[0, 1)$ promotes numerical robustness.

Due to independent lens and subpixel samples, Area ReSTIR cannot rely on *rasterized* G-buffers. As ReSTIR defers shading until after spatiotemporal reuse, to avoid retracing we still store primary hits in a buffer. We also store motion vectors for temporal reuse (in Section 4.3) and depth (for MIS scaling in Section 5.3).

We no longer use Bitterli et al.'s [2020] depth and normal based neighbor rejection heuristic in our Area ReSTIR's spatial reuse implementation, as we found its effect imperceptible when using Lin et al.'s [2022] pairwise MIS. These depth and normal based heuristics are also incompatible with depth of field rendering.

5.2 Ray Tracing Optimization

In ReSTIR, shifting samples *always* requires more visibility tests for correct MIS weights (and thus avoid bias). Area ReSTIR must test visibility $V(\mathbf{x}_0 \to \mathbf{x}_1)$ in every pixel we shift into or any pixel that may have generated a selected sample. As pixels have varying lens and subpixel positions, this needs more visibility tests than prior work. As an example, our fast temporal reuse (Section 4.3.1) requires tracing two primary rays per pixel besides the two shadow rays traced in the original ReSTIR. We need to trace new primary hits, while the original ReSTIR reads them from the G-Buffer.

Our robust temporal reuse shifts four paths to the target pixel (Figure 4, right). For each, three more target function evaluations are needed for MIS weights to account for potential sources. A naive unbiased implementation of our robust temporal reuse (Section 4.3.2), thus requires 12 additional primary rays for each pixel (and 12 shadow rays for direct light).

But pixels only have 8 neighbors. Using a scatter approach, we can exhaustively compute visibility to neighbors' reservoir samples. This requires only 8 primary (and shadow) rays per pixel, rather than gathering visibility for all 16 samples. The eight neighbor rays are also highly coherent; we coalesce tracing to maximize coherent memory accesses during traversal. Compared to a naive approach, this runs 40-60% faster on scenes in our test suite.

5.3 Shift Mappings

Earlier ReSTIR work reuses paths by shift mapping $\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\cdots$ into $\mathbf{x}_0'\mathbf{x}_1'\mathbf{x}_2'\cdots$ with vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ fixed or randomized as needed by the implementation. In Area ReSTIR, it is necessary to explicitly shift map these first two vertices.

In Section 4.6 we described two possible ways, the lens vertex copy and primary hit reconnection shifts.

Neither is always better, so we opt to perform both and combine them via MIS weights. We also weight these strategies via a heuristic modelling their expected quality. This is important, as MIS increases variance when combining a known-good and a bad estimator. We know primary hit reconnections fail for pinhole cameras (or on the focal plane) and the lens copy is less valuable with large circles of confusion. Our heuristic accounts for this domain knowledge.

This heuristic defines a scaling factor γ multiplied in as a confidence weight [Wyman et al. 2023] when computing these shift's MIS weights. $\gamma = 1$ forces selection of the lens vertex copy, and $\gamma = 0$ selects the primary hit reconnection.

We leave theoretical derivation of the scaling factor for future work, and instead derived one empirically. We found image quality somewhat insensitive to the exact values of γ, and we use

$$\gamma(r) = \min\left(1.0, 0.2 + \frac{6.2}{r - 5.6}\right), \tag{13}$$

where r is the circle of confusion diameter in pixels. While resolution-dependent, we tested it works well over varying resolutions (from

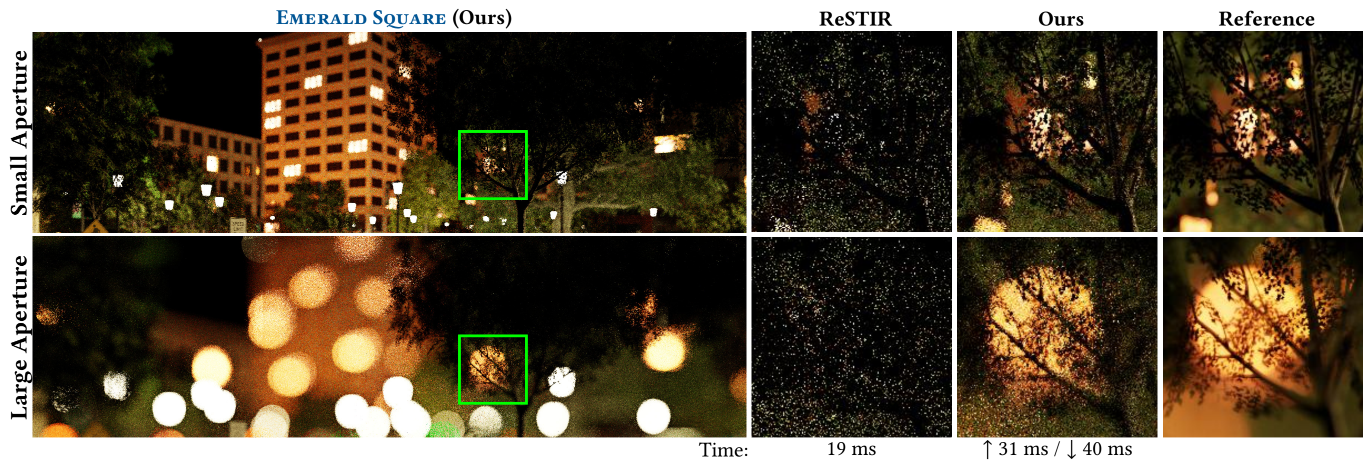


Fig. 7. *Our path reuse works well with various lens apertures, ranging from small to large. We show two very different sized lenses here in* Emerald Square*, but Area ReSTIR outperforms baseline reuse in both cases. These results were captured without camera motion.*

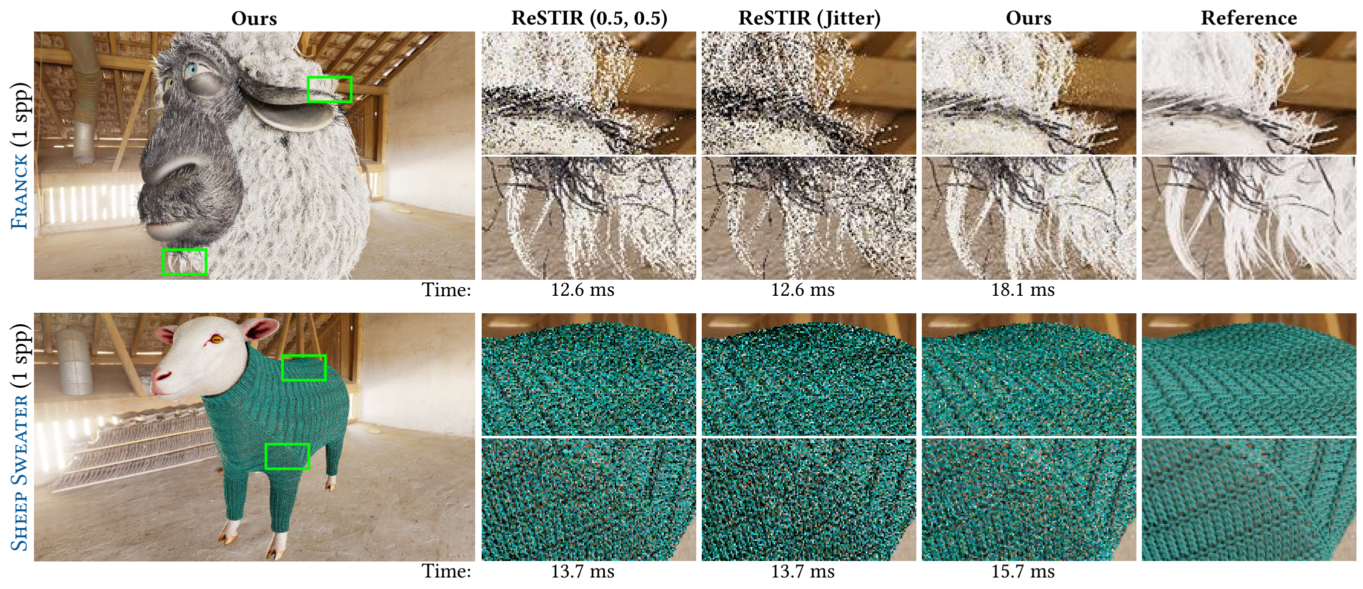


Fig. 8. *When shading pixel centers, ReSTIR gives aliasing. If converged, jittering removes aliasing but does not help quality at 1 spp. Our Area ReSTIR improves temporal reuse, giving higher-quality antialiasing. Note that, based on our MIS scaling function in Section 5.3 (i.e. $\gamma = 1$ for pinhole camera), only* lens vertex copy *shift is applied here. Images captured with a static pinhole camera.* Franck *uses global illumination and ReSTIR PT baseline [Lin et al. 2022].* Sheep Sweater *only uses direct lighting and compares to Bitterli et al. [2020].*

720p to 2160p) and generally found this heuristic handled all scenes well, regardless of aperture size.

When temporally reusing a sample selected from the prior frame, we apply both mappings when shifting back to the current frame (Section 4.3.3), using scaling factor γ as part of our MIS weights. For spatial reuse, we split neighbors into two groups (in a $\gamma : 1-\gamma$ ratio) and each group uses just one type of shift map. We use four spatial neighbors and always assign at least one to use a lens vertex copy.

6 RESULTS

Our prototype builds on the Falcor framework [Kallweit et al. 2022]. Comparisons use the same geometry, lighting, and code framework. We measured performance on an NVIDIA RTX 4090 with an AMD Ryzen 9 7950X and 64GB RAM in resolution 1920 × 1080. Error values are reported using mean absolute percentage error[3] (MAPE). Note that our prototype has no assumptions stipulating any part of the scene is static (e.g., no precomputations on load).

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

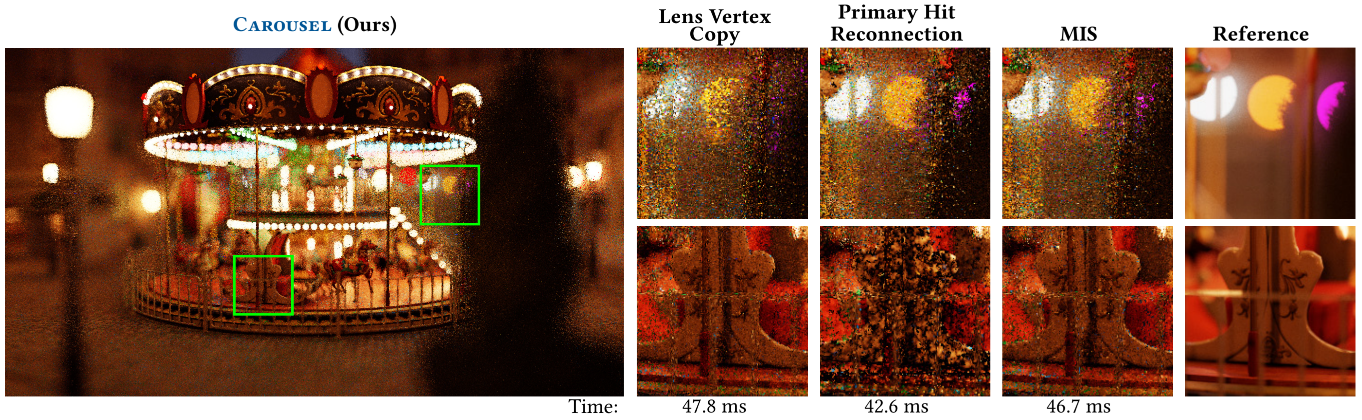


Fig. 9. *The* lens vertex copy *and* primary hit reconnection *shifts (Section 4.6) work better in different situations. The vertex copy works best for regions in focus, while the hit reconnection works better in bokeh and out of focus regions. We MIS between these shift maps (Section 5.3) to achieve good quality everywhere. These* CAROUSEL *results were captured with camera motion, using our robust temporal reuse (Section 4.3.2) and the same shift map (either lens vertex copy or primary hit reconnection) for both spatial and temporal reuse.*

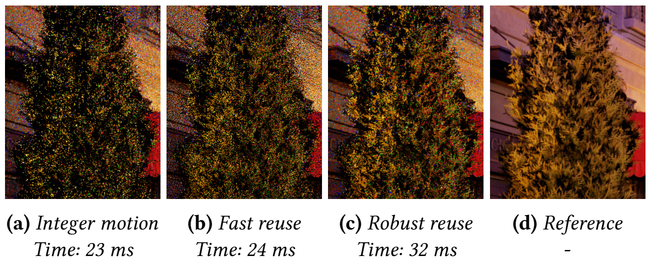


Fig. 10. *Fractional motion vectors enable better reuse across our scenes. Here we compare our method (a) using integer motion vectors to (c) fractional motion vectors with our robust reuse. With fractional motion, our fast reuse (b) also improves quality, but may not find suitable candidates for all pixels. Images captured with camera motion in* BISTRO.

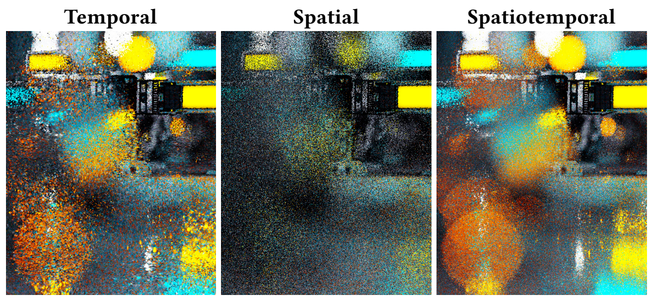


Fig. 11. *Combined spatiotemporal reuse is much more powerful than spatial or temporal reuse alone with Area ReSTIR, especially in areas with strong depth of field. Picture with camera motion in* ZERO DAY.

All ReSTIR variants shown shade just one path per pixel, though many samples contribute via its unbiased contribution weight. We use identical ReSTIR settings between Area ReSTIR and our baseline. For direct lighting, we use 32 light and 1 BSDF candidate samples per pixel [Bitterli et al. 2020]. For path tracing, we use 1 candidate path tree per pixel [Lin et al. 2022]. For temporal reuse, we set history length to 20 (Wyman et al. [2023], Chapter 4.4). For spatial reuse, we reuse 4 neighbors randomly selected in a 30 pixel radius.

Result images were captured either during motion or with a static camera, as listed. Please see our supplementary video for more evaluation under motion.

Improved sample reuse in bokeh. Our new primary hit reconnection shift (Section 4.6) drastically improves sampling of bokeh compared to baseline ReSTIR. Our approach enables hierarchical reuse of bokeh samples, maintaining a well importance sampled set of samples between frames. In Figure 7 we study this benefit for both small and large lens apertures.

Area ReSTIR improves antialiasing and robustness. Prior resampling methods suffer near high subpixel frequencies (see Section 4.2). Figure 8 evaluates our improvements on FRANCK the sheep with mesh geometries, and SHEEP SWEATER with actual curve geometries. Both rendered with a still camera and no depth of field. Shading only pixel centers causes visible aliasing. Jittering removes the aliasing (at high sample counts) but fails to improve quality with just one sample per pixel. Area ReSTIR improves antialiasing and is more robust on those high-frequency fur or fabric.

Robustness with MIS between shift mappings. Since our new primary hit reconnection shift works best for defocused pixels, and the lens vertex copy shift works best for in-focus pixels, we combine the two with MIS (see Section 5.3). We explore the benefits in Figure 9, showing our MIS scaling heuristic works well across pixels with varying levels of focus.

Fractional motion vectors improve robustness. To show the importance of fractional motion, Figure 10 combines different pieces of our work. With area reservoirs and new shift maps, but only integer motion vectors we still have significant noise near edges. Using fractional motion with our fast reuse (Section 4.3.1) improves results but not consistently, as it still resets temporal history when

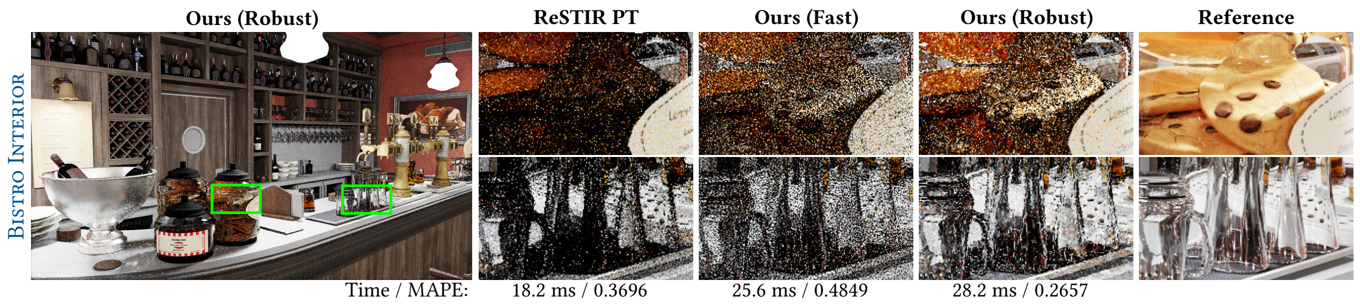


Fig. 12. *Compared to Lin et al.'s [2022] ReSTIR PT, our Area ReSTIR improves path reuse for complex paths like L(S)DSE paths on the cookie jar (top inset) and the L(D)S+E paths for the glasses (bottom) due to more stable temporal reuse and invalidating fewer accumulated sample distributions. While our fast version has higher MAPE than baseline ReSTIR PT, the more difficult, hard-to-denoise paths are sampled significantly better. This scene captured with camera motion.*

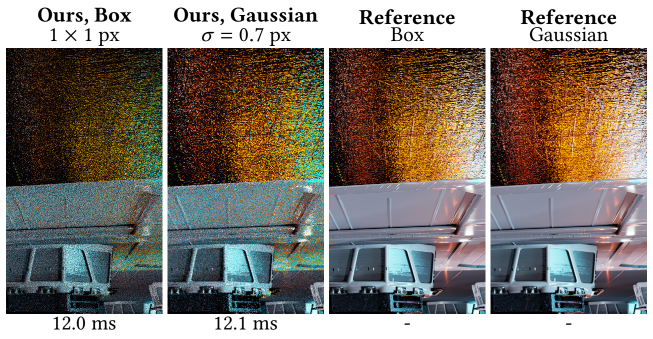


Fig. 13. *Area ReSTIR with our fast reuse benefits from larger pixel filters. Here we compare the 1×1 box filter against a truncated Gaussian with a two-pixel radius and find improvement in reuse quality. Picture with camera motion in* Zero Day.

the reuse fails. Our robust reuse (Section 4.3.2) improves quality further, though at a performance cost.

Spatial and temporal reuse are both important. Prior work found temporal samples contribute most improvement in ReSTIR [Wyman and Panteleev 2021]. While still true, Figure 11 shows combining spatiotemporal samples remains valuable. Spatial reuse helps break up correlations that accumulate during complex nonlinear motion. This is especially true with strong depth of field.

Area ReSTIR improves glossy reflections. Specular paths are highly direction-dependent, which makes reuse tricky, especially under motion. Figure 12 shows our higher dimensional space improves reuse of these tricky paths. This results from our smaller sensitivity to pixel-level details (Section 4.2) and our potential to better reuse nearby paths' primary hit points.

Better reuse with advanced pixel filters. While most of our results use a 1×1 box filter, Section 4.4 extends our work to other pixel filters *without* using kernel splatting. Larger filter supports can improve our temporal reuse, perhaps making such filters attractive for real-time use. We show an example with our fast temporal reuse (Section 4.3.1) in Figure 13.

General performance and quality evaluation. In Figure 15, we evaluate our quality by comparing to prior ReSTIR algorithms [Bitterli et al. 2020; Lin et al. 2022] as a baseline. Our comparisons cover various geometric and lighting scenarios: direct lighting v.s. global illumination; emissive geometry v.s. environment map lighting; diffuse v.s. specular or glossy materals; varying aperture sizes; and high-frequency foliage and normals maps.

Area ReSTIR and baseline ReSTIR use identical independent sample counts and the same number of spatial neighbors, but our area reservoirs allow much better temporal reuse. This gives consistent quality improvement in Figure 15. Our robust temporal reuse (Section 4.3.2) gives better quality than the fast reuse (Section 4.3.1), albeit at higher cost due to more visibility rays for MIS weights. Additionally, running multiple instances of the baseline ReSTIR and averging their results provides very limited improvement in challenging regions, still falling short of the quality achieved by our method at equal time.

Performance overheads grow in scenes with costly ray tracing, e.g., more triangles, complex BVHs, large bokeh with incoherent primary rays, or lots of alpha testing[4]. This overhead comes from tracing rays to compute unbiased MIS weights.

To quantify this ray overhead, we measured our optimized visibility passes (Section 5.2), which coalesce the extra visibility rays into separate ray wavefronts. For the scenes in Figure 15, this overhead was 4.9 ms for Bistro, 4.2 ms for Zero Day, 11.1 ms for Emerald Square, 4.1 ms for Sidewalk, and 4.7 ms for Bistro Interior. This accounts for most overhead moving from our fast to our robust reuse. As correct MIS weights ensure unbiasedness, cheaper alternatives may exist for renderers willing to accept some bias.

Figure 15 includes the Crown scene, using a pinhole camera and thus only antialiasing. Zoom in on the insets to see our antialiasing quality (particularly the robust reuse). In this scene, our fast reuse produces higher total error than the baseline due to failures to find prior-frame samples inside the backprojected filter. This reduces quality in the large in-focus, easy-to-sample areas, but the fewer high-frequency areas are sampled better.

[4]We do not yet use opacity micromaps to improve alpha testing performance.

7 LIMITATIONS AND FUTURE WORK

Overall, using area reservoirs and integrating over additional lens and subpixel dimensions greatly improves quality, both in static images and under motion. However, there are limitations and interesting future directions.

We expect area reservoirs can be extended one more dimension to handle motion blur. But motion blur interacts with the ray acceleration structure in complex ways, especially when aiming to reuse temporal samples. We aimed to solve the challenges of area reservoirs in a simpler domain before adding complex engineering challenges.

As most prior work, we resample a scalar target function. This means we do not importance sample variations between RGB channels; this causes color noise, especially in scenes with many multicolor lights (e.g., the Bistro).

Our assumption that pixels simplistically translate between frames limits reuse quality when zooming or in scenes with more complex motion. Clearly, we miss some relevant samples in integration domains of interest. But it is unclear if the cost of more complex projections and querying arbitrary footprints leads to sufficient reuse benefits. This is worth exploring.

We apply area sampling only during temporal reuse, as spatial neighbors' 1×1 box filters do not have overlapping domains. But for pixel filters with larger domains (e.g., Mitchell-Netravali or Sinc), spatial neighbors may have overlapping footprints. While such filters are typically too costly for real time, future work might find these costs can overlap with better sample reuse.

8 CONCLUSION

We introduced *Area ReSTIR*, a real-time algorithm extending spatiotemporal path reuse to importance sample pixel and aperture footprints. Prior work reuses discretely sampled paths from an image; this misses reusable neighbor paths near high frequencies. By explicitly integrating over image-space, our work finds better reuse candidates. This allows building real-time path tracers that handle bokeh, foliage, hair, and detailed normal maps all with only one shaded sample per pixel (per frame).

While the curse of dimensionality is real, our work shows integrating in higher-dimensional domains gives benefits if it permits more effective reuse than lower dimensions. If ReSTIR better reuses samples, it greatly improves quality.

ACKNOWLEDGMENTS

We want to thank Aaron Lefohn for discussions on and support of this research, and the anonymous reviewers for their valuable comments and feedback. This project was supported in part by NSF Grant #1956085.

A JACOBIAN OF PRIMARY HIT RECONNECTION

Consider shifting path sample $(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ to $(\mathbf{u}', \mathbf{s}', \hat{\mathbf{x}}')$ with our primary hit reconnection, copying $\mathbf{x}_1' = \mathbf{x}_1$. We first acquire $\mathbf{u}'$ as a translation by the locally constant motion vector, so the Jacobian of that part is 1. Next, we want to shift $\mathbf{s}$ to $\mathbf{s}'$, producing some Jacobian $|\partial\mathbf{s}'/\partial\mathbf{s}|$, given $\mathbf{u}'$. Camera vertex $\mathbf{x}_0$ comes from $\mathbf{s}$ by a uniform

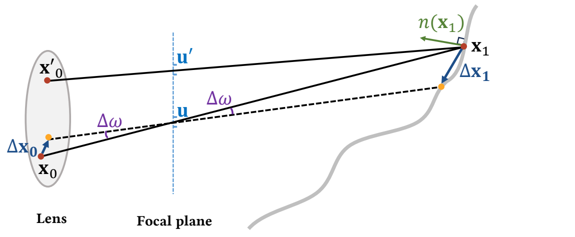


Fig. 14. *Important parameters for deriving the* primary hit reconnection *shift's Jacobian in Appendix A.*

lens sampler, so we have $|\partial\mathbf{x}_0/\partial\mathbf{s}| = (|\partial\mathbf{s}'/\partial\mathbf{x}_0'|)^{-1}$ = lens area. As a result, $|\partial\mathbf{s}'/\partial\mathbf{s}| = |\partial\mathbf{x}_0'/\partial\mathbf{x}_0|$. Thus, we must just find $|\partial\mathbf{x}_0'/\partial\mathbf{x}_0|$.

With $\mathbf{u}'$ acquired, reusing $\mathbf{x}_1$ gives a new $\mathbf{x}_0' \neq \mathbf{x}_0$ (Figure 14), and

$$\left|\frac{\partial\mathbf{x}_0'}{\partial\mathbf{x}_0}\right| = \left|\frac{\partial\mathbf{x}_0'}{\partial\mathbf{x}_1}\right|\left|\frac{\partial\mathbf{x}_1}{\partial\mathbf{x}_0}\right|. \tag{14}$$

We must figure how $\mathbf{x}_0$ and $\mathbf{x}_0'$ change with $\mathbf{x}_1$, given fixed $\mathbf{u}$ and $\mathbf{u}'$. Figure 14 shows a solid angle change $\Delta\boldsymbol{\omega}$ at the focal plane (corresponding to $\mathbf{u}$) changes $\Delta\mathbf{x}_0$ and $\Delta\mathbf{x}_1$ differently. We can apply the solid-angle to area measure Jacobian to get

$$\left|\frac{\partial\mathbf{x}_1}{\partial\mathbf{x}_0}\right| = \frac{d_1^2}{|\boldsymbol{\omega}\cdot n(\mathbf{x}_1)|}\frac{|\boldsymbol{\omega}\cdot n(\mathbf{x}_0)|}{d_0^2}\,, \tag{15}$$

given the direction $\boldsymbol{\omega}$ along $\overline{\mathbf{x}_0\mathbf{x}_1}$, lens and surface normals n, and distances d_0 and d_1 between the focal plane and $\mathbf{x}_0$, $\mathbf{x}_1$. Similarly,

$$\left|\frac{\partial\mathbf{x}_0'}{\partial\mathbf{x}_1}\right| = \left|\frac{\partial\mathbf{x}_0'}{\partial\boldsymbol{\omega}'}\right|\left|\frac{\partial\boldsymbol{\omega}'}{\partial\mathbf{x}_1}\right| = \frac{d_0'^2}{|\boldsymbol{\omega}'\cdot n(\mathbf{x}_0')|}\frac{|\boldsymbol{\omega}'\cdot n(\mathbf{x}_1)|}{d_1'^2}\,, \tag{16}$$

where $\boldsymbol{\omega}'$, d_0', d_1' are similarly defined with $\mathbf{x}_1$, $\mathbf{x}_0'$, and $\mathbf{u}'$. Note that $n(\mathbf{x}_0')$ can vary from $n(\mathbf{x}_0)$ if the lens moves.

Given $\mathbf{u}'$ and $\mathbf{s}'$, we can now finish the full shift by acquiring $\hat{\mathbf{x}}'$ from $\hat{\mathbf{x}}$ by any existing path shift, such as Lin et al.'s [2022] hybrid shift. The full $(\mathbf{u}, \mathbf{s}, \hat{\mathbf{x}})$ is shifted by concatenating the three partial shifts, and hence the full Jacobian is the product of the three Jacobians.

REFERENCES

Guillaume Abadie. 2018. Life of a bokeh. In *SIGGRAPH Course Notes: Advances in Real-Time Rendering*. https://advances.realtimerendering.com/s2018/index.htm

Kurt Akeley. 1993. Reality Engine graphics. In *Proceedings of SIGGRAPH*. https://doi.org/10.1145/166117.166131

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 Trans. Graph.* 36, 4, Article 97 (Jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073708

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain path reusing. *ACM Trans. Graph.* 36, 6, Article 229 (Nov 2017), 9 pages. https://doi.org/10.1145/3130800.3130886

Philippe Bekaert, Mateu Sbert, and John H Halton. 2002. Accelerating path tracing by re-using paths. In *Rendering Techniques*. 125–134. https://doi.org/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 (Jul 2013), 18 pages. https://doi.org/10.1145/2487228.2487239

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 Trans. Graph.* 39, 4, Article 148 (Jul 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

Guillaume Boissé. 2021. World-space spatiotemporal reservoir reuse for ray-traced global illumination. In *SIGGRAPH Asia Technical Communications.* 22:1–4. https://doi.org/10.1145/3478512.3488613

Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. Rendering many lights with grid-based reservoirs. In *Ray Tracing Gems II*, Adam Marrs, Peter Shirley, and Ingo Wald (Eds.). APress, 351–365. https://doi.org/10.1007/978-1-4842-7185-8_23

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for differentiable and inverse rendering. In *SIGGRAPH (Conference Track).* 1–10. https://doi.org/10.1145/3588432.3591512

Min-Te Chao. 1982. A general purpose unequal probability sampling plan. *Biometrika* 69, 3 (1982), 653–656. https://doi.org/10.2307/2336002

Robert L. Cook, Thomas K. Porter, and Loren C. Carpenter. 1984. Distributed ray tracing. *Proceedings of SIGGRAPH.* https://doi.org/10.1145/964965.808590

Joe Demers. 2004. Depth of field: A survey of techniques. In *GPU Gems.* Addison-Wesley, Chapter 23, 375–390.

Cass Everitt. 2001. Interactive order-independent transparency. White Paper, NVIDIA.

Paul Haeberli and Kurt Akeley. 1990. The accumulation buffer: Hardware support for high-quality rendering. In *Proceedings of SIGGRAPH.* https://doi.org/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* 39, 2 (May 2020), 147–155. https://doi.org/10.1111/cgf.13919

Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 2019. A survey on gradient-domain rendering. *Computer Graphics Forum* 38, 2 (2019), 455–472. https://doi.org/10.1111/cgf.13652

Jie Jiang, Xiang Xu, and Beibei Wang. 2023. A ReSTIR GI Method Using the Sample-Space Filtering. In *Computer Graphics International.* 79–92. https://doi.org/10.1007/978-3-031-50078-7_7

Jorge Jimenez. 2014. Next generation post processing in "Call of Duty: Advanced Warfare". In *SIGGRAPH Course Notes: Advances in Real-Time Rendering.* https://advances.realtimerendering.com/s2014/index.html

Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, and Tiago Sousa. 2011. Filtering approaches for real-time anti-aliasing. In *SIGGRAPH Course Notes.* https://www.iryoku.com/aacourse/

James T Kajiya. 1986. The rendering equation. In *Proceedings of SIGGRAPH.* 143–150. https://doi.org/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

Brian Karis. 2014. High-quality temporal supersampling. In *SIGGRAPH Course Notes: Advances in Real-Time Rendering.* https://advances.realtimerendering.com/s2014/index.html

Michael Kass, Aaron Lefohn, and John Owens. 2006. Interactive depth of field using simulated diffusion on a GPU. Pixar Technical Memo, #06-01.

Markus Kettunen, Daqi Lin, Ravi Ramamoorthi, Thomas Bashford-Rogers, and Chris Wyman. 2023. Conditional resampled importance sampling and ReSTIR. In *SIGGRAPH Asia (Conference Track).* https://doi.org/10.1145/3610548.3618245

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain path tracing. *ACM Trans. Graph.* 34, 4, Article 123 (Jul 2015), 13 pages. https://doi.org/10.1145/2766997

Jonathan D. Korein and Norman I. Badler. 1983. Temporal anti-aliasing in computer generated animation. *Proceedings of SIGGRAPH*, 377–388. https://doi.org/10.1145/800059.801168

M. Kraus and M. Strengert. 2007. Depth-of-field rendering by pyramidal image processing. *Computer Graphics Forum* 26, 3 (2007), 645–654. https://doi.org/10.1111/j.1467-8659.2007.01088.x

Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. 2009. Depth-of-field rendering with multiview synthesis. *ACM Trans. Graph.* 28, 5 (Dec 2009), 1–6. https://doi.org/10.1145/1618452.1618480

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain Metropolis light transport. *ACM Trans. Graph.* 32, 4, Article 95 (Jul 2013), 12 pages. https://doi.org/10.1145/2461912.2461943

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.* 31, 6, Article 194 (Nov 2012), 9 pages. https://doi.org/10.1145/2366145.2366213

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: Foundations of ReSTIR. *ACM Trans. Graph.* 41, 2, Article 75 (Aug 2022), 23 pages. https://doi.org/10.1145/3528223.3530158

Daqi Lin, Chris Wyman, and Cem Yuksel. 2021. Fast volume rendering with spatiotemporal reservoir resampling. *ACM Trans. Graph.* 40, 6, Article 279 (Dec 2021), 18 pages. https://doi.org/10.1145/3478513.3480499

Timothy Lottes. 2009. FXAA. White Paper, NVIDIA.

Don P. Mitchell and Arun N. Netravali. 1988. Reconstruction filters in computer-graphics. *SIGGRAPH Comput. Graph.* 22, 4 (Jun 1988), 221–228. https://doi.org/10.1145/378456.378514

NVIDIA. 2019. Deep learning super sampling. https://developer.nvidia.com/rtx/dlss [Online; accessed 15-January-2024].

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path resampling for real-time path tracing. In *Computer Graphics Forum*, Vol. 40. 17–29. https://doi.org/10.1111/cgf.14378

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann, Cambridge, MA. https://pbrt.org/

Michael Potmesil and Indranil Chakravarty. 1982. Synthetic image generation with a lens and aperture camera model. *ACM Trans. Graph.* 1, 2 (Apr 1982), 85–108. https://doi.org/10.1145/357299.357300

Alexander Reshetov. 2009. Morphological antialiasing. In *High Performance Graphics.* 109–116. https://doi.org/10.1145/1572769.1572787

Rohan Sawhney, Daqi Lin, Markus Kettunen, Benedikt Bitterli, Ravi Ramamoorthi, Chris Wyman, and Matt Pharr. 2024. Decorrelating restir samplers via mcmc mutations. *ACM Trans. Graph.* 43, 1 (2024), 1–15. https://doi.org/10.1145/3629166

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance resampling for global illumination. In *Eurographics Symposium on Rendering.* 139–146. https://doi.org/10.2312/EGWR/EGSR05/139-146

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph. D. Dissertation. Stanford, CA, USA. https://dl.acm.org/doi/10.5555/927297

Eric Veach and Leonidas Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of SIGGRAPH.* 419–428. https://doi.org/10.1145/218380.218498

Eric Veach and Leonidas J. Guibas. 1997. Metropolis light transport. In *Proceedings of SIGGRAPH.* 65–76. https://doi.org/10.1145/258734.258775

Yu-Chen Wang, Chris Wyman, Lifan Wu, and Shuang Zhao. 2023. Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR. *ACM Trans. Graph.* 42, 6 (December 2023), 1–17. https://doi.org/10.1145/3618331

Alexander Weinrauch, Wolfgang Tatzgern, Pascal Stadlbauer, Alexis Crickx, Jozef Hladky, Arno Coomans, Martin Winter, Joerg H. Mueller, and Markus Steinberger. 2023. Effect-based multi-viewer caching for cloud-native rendering. *ACM Trans. Graph.* 42, 4, Article 87 (Jul 2023), 16 pages. https://doi.org/10.1145/3592431

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 Courses.* https://doi.org/10.1145/3587423.3595511

Chris Wyman and Alexey Panteleev. 2021. Rearchitecting spatiotemporal resampling for production. In *High-Performance Graphics.* 23–41. https://doi.org/10.2312/hpg.20211281

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Trans. Graph.* 39, 4, Article 142 (Aug 2020), 12 pages. https://doi.org/10.1145/3386569.3392376

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of temporal antialiasing techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. https://doi.org/10.1111/cgf.14018

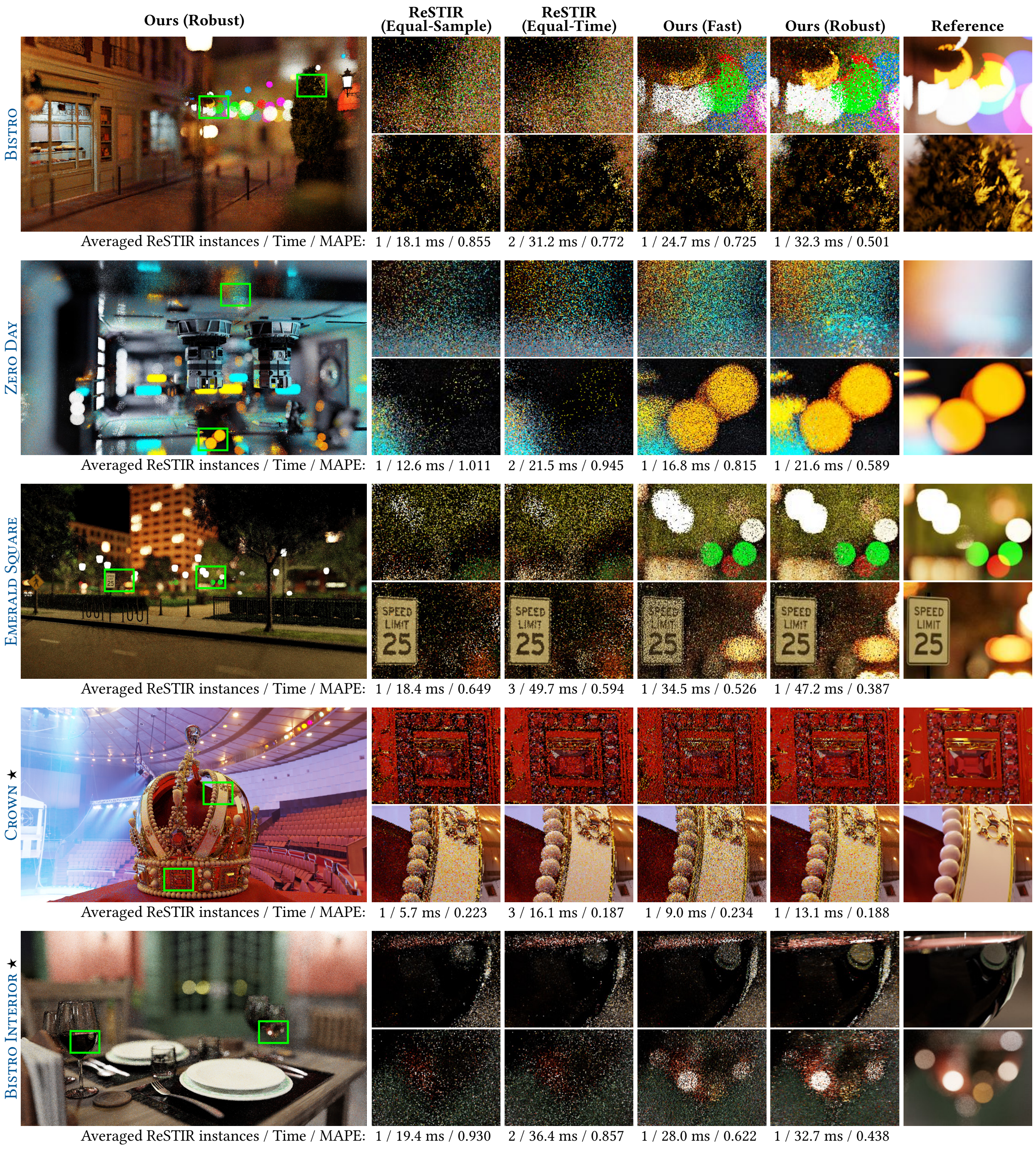

Ours (Robust)
ReSTIR (Equal-Sample)
ReSTIR (Equal-Time)
Ours (Fast)
Ours (Robust)
Reference
Bistro
Averaged ReSTIR instances / Time / MAPE: 1 / 18.1 ms / 0.855 2 / 31.2 ms / 0.772 1 / 24.7 ms / 0.725 1 / 32.3 ms / 0.501
Zero Day
Averaged ReSTIR instances / Time / MAPE: 1 / 12.6 ms / 1.011 2 / 21.5 ms / 0.945 1 / 16.8 ms / 0.815 1 / 21.6 ms / 0.589
Emerald Square
Averaged ReSTIR instances / Time / MAPE: 1 / 18.4 ms / 0.649 3 / 49.7 ms / 0.594 1 / 34.5 ms / 0.526 1 / 47.2 ms / 0.387
Crown ★
Averaged ReSTIR instances / Time / MAPE: 1 / 5.7 ms / 0.223 3 / 16.1 ms / 0.187 1 / 9.0 ms / 0.234 1 / 13.1 ms / 0.188
Bistro Interior ★
Averaged ReSTIR instances / Time / MAPE: 1 / 19.4 ms / 0.930 2 / 36.4 ms / 0.857 1 / 28.0 ms / 0.622 1 / 32.7 ms / 0.438

Fig. 15. *Comparing our Area ReSTIR with prior ReSTIR baselines. Starred (★) scenes use global illumination and a ReSTIR PT baseline [Lin et al. 2022], other scenes use only direct lighting and compare to Bitterli et al. [2020]. Except the second column, we use the same number of independent samples within each comparison. In the second column, we try to match baseline ReSTIR's rendering time to our robust version by running multiple independent copies of ReSTIR and averaging their results. All scenes were captured under camera motion. Performance numbers are averaged over a 10 second animation.*