Supplemental Document: Conditional Resampled Importance Sampling and ReSTIR

Markus Kettunen*
NVIDIA
Finland
mkettunen@nvidia.com

Daqi Lin*
NVIDIA
USA
daqil@nvidia.com

Ravi Ramamoorthi
NVIDIA and UC San Diego
USA
ravir@cs.ucsd.edu

Thomas Bashford-Rogers
University of Warwick
UK
thomas.bashford-rogers@warwick.ac.uk

Chris Wyman
NVIDIA
USA
chris.wyman@acm.org

ACM Reference Format:
Markus Kettunen, Daqi Lin, Ravi Ramamoorthi, Thomas Bashford-Rogers, and Chris Wyman. 2023. Supplemental Document: Conditional Resampled Importance Sampling and ReSTIR. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3610548.3618245

This document provides additional information about our conditional ReSTIR suffix reuse and final gather prototype (Section S.1), as well as additional experiments (Section S.2), notes on performance (Section S.3), convergence (Section S.4), and an extended pseudocode with technical considerations (Section S.5). We also discuss quality limitations of our current final gather prototype (Section S.6).

Please note: Experiments and details are provided for validation and as a potential starting point for future experiments; we do not expect our prototype, in its current form, to become widely used in real applications.

S.1 THEORY AND IMPLEMENTATION DETAILS

In this section, we provide more details about our prototype and some parts of the theory.

S.1.1 When is a Candidate Valid for Conditional ReSTIR

Reusing conditioned samples imposes some restrictions on the allowed dependencies between the conditioning and the conditioned random variables—a wrong dependency may make using a sample as a candidate impossible.

Tricky scenarios arise if the conditional context depends on the candidate samples, making a candidate's unbiased contribution weight invalid in the correlated conditional context. But not all dependencies are forbidden, only certain kind of conditional dependencies.

For example, picking new prefixes with target function dependent on prior suffixes biases future computations, i.e., any future resampled suffixes conditioned on these prefixes. Intuitively, view such circular dependencies as bad. But one might ask: how can driving supporting prefixes with the suffixes invalidate future *suffixes*? After all, supporting prefixes are never directly used for integration. We pick independent prefixes for shading, and merely choose the suffix reservoir by a k-NN search among the supporting prefixes.

Let X^p, X^s be the previous frame's prefix and suffix with contribution weights W_{X^p} and $W_{X^s|X^p}$. To use this candidate to select next frame's suffix Y^s with conditional ReSTIR, given next frame's prefix Y^p, we need a contribution weight $W_{X^s|X^p,Y^p}$. However, if Y^p is updated dependently to X^s, it is no longer conditionally independent of X^s and $W_{X^s|X^p}$ (given X^p), and hence $W_{X^s|X^p}$ can no longer be used as a drop-in replacement for $W_{X^s|X^p,Y^p}$, required for conditional ReSTIR. The conditional dependency between the new prefix Y^p and the old suffix X^s makes the new context and the old sample incompatible.

The following theorem formally explains the conditional independence requirement:

> Theorem S.1. *Let X be a random variable with UCW W_X, and let X and W_X be independent of Z. Then $W_{X|Z} := W_X$ is a conditional UCW for X, given Z.*

Proof. Let f be any integrable function of x and z. By independence of both X and W_X from Z,

$$\mathbb{E}\left[f(X,Z)W_{X|Z}|Z\right] = \mathop{\mathbb{E}}_{X,W_X}\left[f(X,Z)W_X\right] \tag{S.1}$$

$$= \int_{\mathrm{supp}(X)} f(x,Z)\,\mathrm{d}x \tag{S.2}$$

$$= \int_{\mathrm{supp}(X|Z)} f(x,Z)\,\mathrm{d}x, \tag{S.3}$$

proving by definition (Equation 5) that W_X is a conditional UCW for X, given Z. □

*Joint first authors; equal contribution.

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12.
https://doi.org/10.1145/3610548.3618245

Theorem S.2. *Let X be a random variable with conditional UCW $W_{X|Y}$, and let X and $W_{X|Y}$ be independent of Z, given Y. Then $W_{X|Y,Z} := W_{X|Y}$ is a conditional UCW for X, given Y and Z.*

Proof. Apply the previous theorem to X and $W_{X|Y}$ in the probability space conditioned by Y. □

Less formally, we can add a random variable Z to condition W_X, if Z is independent of X and W_X, given the previous conditioning variables.

By substituting the previous frame's suffix X^s for X, all previous supporting prefixes as vector X^p for Y, and the new frame's supporting prefixes as vector Y^p for Z, we read:

Let the previous suffix X^s have conditional UCW $W_{X^s|X^p}$, and let the new supporting prefixes Y^p be independent of both X^s and its UCW $W_{X^s|X^p}$, given old prefixes X^p. Then, $W_{X^s|X^p}$ can be used as $W_{X^s|X^p,Y^p}$ when passing X^s into conditional ReSTIR in the next frame.

The suffixes and prefixes must be kept independent, except for the shared context of the old supporting prefixes.

S.1.2 Forbidden Conditional Independence in Joint UCWs

With certain constraints, the product of two UCWs W_{X_1} and W_{X_2} can form a joint UCW W_{X_1,X_2}. The simplest case is when W_{X_1} and W_{X_2} are independent, but this is often a very demanding requirement. Often, we want to join a marginal UCW W_{X_1} with a conditional UCW $W_{X_2|X_1}$, which calls for a weaker constraint than total independence.

As described in Theorem 4.1, the required constraint is conditional independence: X_2 and $W_{X_2|X_1}$ must be conditionally independent of W_{X_1}, given X_1. They all can depend on the shared context X_1, but other than that, X_2 and $W_{X_2|X_1}$ must be independent of W_{X_1}.

Imagine a simplified two-pass ReSTIR where we update a prefix X^p with a target function depending on a suffix X^s, applying ReSTIR, and vice versa for the suffix, with conditional ReSTIR. Assume we somehow acquire valid UCWs W_{X^p} and $W_{X^s|X^p}$ (the conditional dependency makes this hard, see Section S.1.1.) Still, this back-and-forth ReSTIR would lead to bias when evaluating $\langle I \rangle = f(X^p, X^s) W_{X^p} W_{X^s|X^p}$, since $W_{X^p} W_{X^s|X^p}$ does not form a joint UCW W_{X^p,X^s} due to the conditional dependency between W_{X^p} and X^s (and $W_{X^s|X^p}$), given X^p.

S.1.3 Interpretation of Domain Weights

The generalized balance heuristic (Equation 15),

$$m_i(y) = \frac{\alpha_i \, \hat{p}_{\leftarrow i}(y)}{\sum_{j=1}^{M} \alpha_j \, \hat{p}_{\leftarrow j}(y)}, \tag{S.4}$$

uses Jacobian-corrected target functions $\hat{p}_{\leftarrow}$ as proxies for unknown (conditional) PDFs. Assuming the proxy PDFs were exactly proportional to the unknown PDFs, i.e., if we had

$$\frac{\hat{p}_{\leftarrow j}}{\|\hat{p}_{\leftarrow j}\|_1} = p_j, \tag{S.5}$$

then we could rewrite the generalized balance heuristic as

$$m_i(y) = \frac{(\alpha_i \, \|\hat{p}_{\leftarrow i}\|_1) \; p_i(y)}{\sum_{j=1}^{M} (\alpha_j \, \|\hat{p}_{\leftarrow j}\|) \, p_j(y)}. \tag{S.6}$$

Comparing this to Veach and Guibas' [1995] multi-sample MIS,

$$m_i(y) = \frac{n_i \, p_i(y)}{\sum_{j=1}^{M} n_j \, p_j(y)}, \tag{S.7}$$

which combines $n_1 + \cdots + n_M$ samples each for strategies $1, \ldots, M$, we present a heuristic argument. If each sample Y_i were equivalent to n_i independent samples (i.e., n_i is the effective sample count), then we should pick the α_i such that $\alpha_i \, \|\hat{p}_{\leftarrow i}\|_1 = n_i$, making the generalized balance heuristic effectively into multi-sample MIS.

The norms $\|\hat{p}_{\leftarrow i}\|_1$ are generally not known. This is non-issue when all $\hat{p}_{\leftarrow j}$ have approximately the same 1-norm, as the norms cancel out, and we can match multi-sample MIS by selecting $\alpha_i = n_i$, the effective sample count, or confidence weight, of the input sample.

If the norms $\|\hat{p}_{\leftarrow i}\|_1$ vary, we could, in principle, avoid a variance increase by choosing $\alpha_i = n_i / \|\hat{p}_{\leftarrow i}\|_1$, but the $\|\hat{p}_{\leftarrow i}\|_1$ are normally not known. Hence, if we can bring the norms closer by other means, we should. In our case, we drop the unnecessary factor $f_p(x^p)$ from the suffix's target function in suffix ReSTIR; this does not change the suffix's distribution, but it decreases implicit domain weighting.

Bitterli et al. [2020] presented a similar argument in their supplemental document; we expand on it by making the connection to multi-sample MIS and taking into account effective sample counts.

S.1.4 Domain Weights in Our Prototype

Adhering to the previous section, we set the domain weights α_i for prefix resampling (Algorithm 1, line 5), suffix resampling (Algorithm 1, line 7), and integration (Algorithm 1, lines 12 and 16) proportional to the reservoirs' effective sample counts, i.e., the confidence weight M_r (Section 6.2 in [Lin et al. 2022]).

A new sample always gets $M_r = 1$, while the suffix reservoirs' M_r increases through spatiotemporal reuse up to a cap of $M_r^{\max} = 50$. Capping M_r is necessary [Lin et al. 2022], and it avoids excessive temporal correlation and potential sample impoverishment. Our value of $M_r^{\max} = 50$ is greater than $M_r^{\max} = 20$ used in prior work, as our final gather effectively removes visible correlations; a higher $M_r^{\max}$ allows us to benefit more from temporal accumulation. We observe diminishing returns from $M_r^{\max}$ larger than 50.

For the spatial reuse pass for ReSTIR suffixes (Algorithm 1, line 8), we use 3 neighbors within a 30-pixel screen space radius and MIS with the *defensive variant* of the generalized pairwise MIS [Bitterli 2021; Lin et al. 2022]. Pairwise MIS favors the center pixel's own sample, as it is already in the right domain and thus tends to be more suitably distributed than spatial neighbors.

S.1.5 Spatial Neighbor Search

At integration time, having sampled an integration prefix, we search for the closest supporting prefixes to borrow their ReSTIR suffixes. While more advanced heuristics would likely lead to better results, our proof-of-concept simply compares the world-space locations of the prefixes' last vertices.

To find the closest supporting prefixes, we build a BVH over the supporting prefixes' last vertices to perform approximate fixed-maximum-radius k-NN queries. The size of the BVH is proportional to the number of screen pixels. We use the algorithm of Evangelou et al. [2021], utilizing hardware ray tracing. We use the DXR API for automatic tree construction and perform hardware accelerated ray queries in shaders. However, this search was not designed nor meant to be optimal. Currently, BVH build and search contributes roughly 30% of our prototype's total cost.

The densities of the supporting prefixes' last vertices tend to vary by region in the world space, partially because they originate from screen space (in a stratified pattern) and spread out. Thus, we use an adaptive search radius for the k-NN queries. In the BVH, each vertex is assigned a bounding box with half-width r given by a certain heuristic formula that scales by the path's Euclidean length:

$$r = 2.0 \times \|\bar{x}\| \times \alpha_{\text{fov}}. \tag{S.8}$$

Here, $\|\bar{x}\|$ is the sum of the lengths of the path segments and α_{fov} is the angular coverage of the screen's central pixel in radians. This heuristic formula is simplistic but good enough for preliminary results with our proof-of-concept implementation; it reasonably well adapts to different scene sizes and viewing positions. We use the same formula for determining the search radius for the k-NN search at an integration prefix's last vertex. If a given number of neighbors cannot be found by k-NN search, we pick random reservoirs in the 30-pixel screen space neighborhood to reuse their suffixes as a fallback.

While path footprints [Bekaert et al. 2003] can more accurately adapt to material types and viewing angles, their cosine terms and scatter probability densities can create highly non-uniform bounding box sizes, making it hard to choose a constant factor that achieves both good performance and quality. However, our ad-hoc formula is merely a first step towards an efficient method for finding good supporting prefixes, which we have shown to be an important problem.

S.1.6 Splitting Paths into Prefixes and Suffixes

Our suffix ReSTIR and final gather integrates over the path space by dividing paths into prefixes and suffixes. To avoid complications, we deterministically divide so that, for a given a path, we always split it the same way; the split of a path x into (x^p, x^s) is unique. This avoids, for instance, double-counting parts of path space. So what is a good way to define a prefix?

Traditional photon mapping sends final gather rays from the first sufficiently rough vertex [Pharr et al. 2016]; these gather rays then hit either rough or glossy surfaces, where the photon mapper then performs radiance estimation. In our terminology, its prefixes end at the vertex after the first rough vertex. If the last vertex is glossy, the final gather ray hits a glossy surface, and has low probability of getting a useful contribution. In our prototype, the same problem exists if we allow prefixes to terminate at a glossy vertex. Thus, we require prefixes to end at a rough vertex.

In order for our final gather to generate decorrelated samples, we need at least one sufficiently rough vertex before reconnecting to ReSTIR-selected suffixes; we need at least two rough vertices. (Paths with only highly glossy and specular interfaces are already highly correlated, spatially and temporally, so only-specular prefixes cannot hide correlations in the suffixes they connect to.)

Our prefix ReSTIR applies shift maps when reusing prefixes between frames. To avoid invalidating suffixes, it is important that the last vertices of the prefixes do not move in the shift mappings; they will only be moved by ReSTIR accepting a new prefix. With the manifold exploration shift mapping [Jakob and Marschner 2012] we could achieve this by defining a prefix to end at the second rough vertex. However, local shift mappings [Kettunen et al. 2015] such as the hybrid shift [Lin et al. 2022] require two consecutive rough vertices to execute a reconnection.

We additionally require that the last segment of the prefix is not too short. This discourages supporting prefixes from clustering near object edges and corners, which would lead to a small set of suffix reservoirs being overused, boosting correlation.

This gives us an improved division of a path into a prefix and a suffix: the prefix is the shortest subpath from the camera that contains two consecutive sufficiently rough vertices that are sufficiently far from each other. The remainder of the path is the suffix.

Figure S.1 compares the quality of our final gather with photon mapping style prefix definition, and our new prefix definition. Our new definition significantly improves image quality at reflections and near object edges.

We define rough vertices as material roughness being at least 0.2 (for the current BSDF lobe, see Section S.1.8). We set the distance threshold to 1% of the scene size; a more robust method is future work. Our average bounce count for integration prefixes is still close to one in most scenes. Figure S.2 shows bounce count maps in two scenes.

S.1.7 Shift Mappings

Our roughness and distance conditions are defined similarly to Lin et al. [2022] and we use their hybrid shift implementation (random replay + reconnection) for temporal prefix reuse. Our suffix reuse also uses the hybrid shift, but conditioned with the prefixes, as mentioned in Section 6.

S.1.8 Multi-layer Materials

Vertices on single-lobe materials are often relatively easy to classify as either rough or smooth, but classification on multi-layer materials is harder: it depends on the sampled BSDF lobe [Lin et al. 2022]. Applying this to a prefix's last vertex comes with a challenge: its material classification then depends on the current ReSTIR-driven suffix. If the first suffix vertex sampled a smooth BSDF lobe, the prefix's last vertex must also turn smooth, making the prefix fail our definition (from Section S.1.6). While this conflict is likely solvable, we leave it for future work.

For simplicity, we only require the last vertex of the prefix to *contain* a rough BSDF component, rather than force sampling it. As a result of this simplification, some subpaths are defined as prefixes at glossy surfaces, causing inefficient suffix reuse. This quality loss is partially compensated by use of a hybrid shift during suffix reuse, which often allows reuse through glossy vertices, but avoiding the issue entirely would be better.

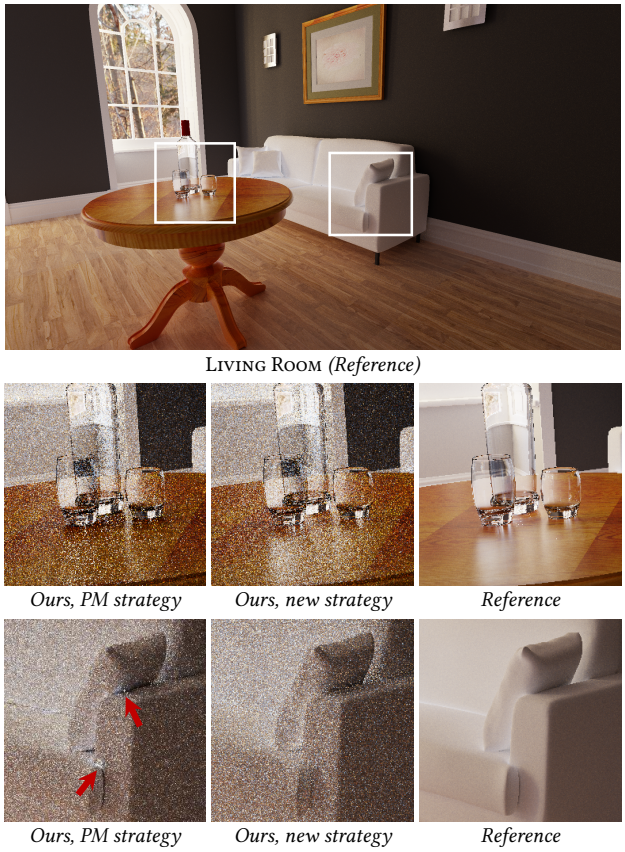

Figure S.1: *Comparing final gather with a photon mapping-inspired prefix definition ("PM") against our improved definition in a 32-integration-prefix gather in the Living Room scene (indirect illumination only). Photon mapping shoots final gather rays at the first diffuse surface of a path. Using the corresponding idea to determine the prefix leads to suffix connections on specular surfaces (like the glasses in the top row's insets), which results in poor reuse. Our improved prefix definition extends prefix paths through specular objects, which shows clear improvement in the caustics on the table. We additionally require prefixes to extend through corners and edges, which removes VPL-like artifacts due to correlation in neighbor search (red arrows). Our new strategy only increases ray count by 0.5%.*

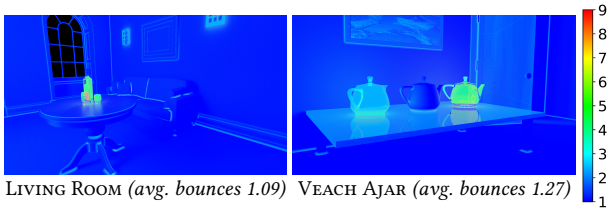

Figure S.2: *Average integration prefix length remains close to one with our improved prefix definition. As shown in these bounce heatmaps, most pixels still have one-bounce prefixes (vertex count = 2 + bounce count). Pixels around corners and edges, or those containing a specular or transmissive material, have longer prefix lengths. Black pixels do not hit any surface.*

S.2 ADDITIONAL ABLATION STUDIES

Figures S.3 and S.4 study the effect on integration variance of the integration prefix count and number of reused ReSTIR suffixes per prefix. In both figures, we show results with just one canonical suffix per pixel (left), implementing our final gather roulette, but we also show results without roulette (right), which samples one canonical suffix for each prefix. Each data point is collected from an independent run and the reservoirs are warmed up before error computation (usually warming up for $M_r^{\max}$ frames will fill the temporal history, see Section S.1.4, though somewhat less is often sufficient).

As shown in the plots, increasing the integration prefix count more effectively decreases variance than increasing the reused suffixes count; prefixes help more and more can be used before encountering diminishing returns. On the other hand, reusing more than a few ReSTIR-suffixes per prefix encounters diminishing returns. This is likely because ReSTIR suffixes of close-by supporting prefixes tend to share information due to spatial reuse.

When using Russian roulette with one canonical suffix for all prefixes (left plots), the task of the integration prefixes is to gather the information stored in the suffixes. This information is high-quality but not unlimited, and eventually increasing integration prefixes should yield diminishing returns. We see this in Zero Day (Figure S.4); a part of the path space is not covered by the ReSTIR suffixes, and while borrowing more suffixes can reduce this region, eventually more canonical suffixes are needed to further decrease variance. In Veach Ajar (Figure S.3), only a small part of the path space is not covered by the ReSTIR suffixes, which very accurately capture the illumination, so improving the final gather with more prefixes, without adding more canonical suffixes, is very effective.

S.3 PERFORMANCE

The current performance is bottlenecked by tracing integration prefixes and searching spatial neighbors. See Table S.1 for an analysis of frame time for Figure 6. As prefix count increases, the final gather cost becomes dominant, whereas the cost of updating the suffix pool is almost constant. This indicates that future optimization effort should focus on reducing the final gather cost or the number of integration prefixes required (by improving their sample distribution). We also provide timings for Figure 3 and 4 (see Table S.2 and S.3). Table S.2 demonstrates the quadratic time complexity of balance heuristic in suffix reuse. Optimization may want to explore biased variants of suffix reuse to reduce the cost of MIS weight computation. In Table S.3, our method with 128 integration prefixes with Russian roulette suffix generation ("128") is about 2× faster to render than the version with a canonical suffix for each prefix ("128 + C") with current implementation, which is a smaller factor compared to the 4-5 × fewer rays shot. With optimization in spatial neighbor search, this gap should become smaller.

Table S.4 gives timings for Figure 1 and 7.

S.4 CONVERGENCE

Figure S.5 and S.6 study the convergence of our algorithm in different scenes using SMAPE (symmetric mean absolute percentage

error)[1] and relMSE (relative mean squared error)[2], by accumulating 256 independent renders of Figure 7. While our method provides lower visual error and lower SMAPE numbers than ReSTIR PT with small number of frames accumulated, it can have higher error numbers than ReSTIR PT with large number of accumulated frames. This is largely caused by fireflies in our Russian roulette algorithm, which the L1 SMAPE numbers tend to filter (which matches the perceived error closer). This suggests that adding more canonical suffixes can help reduce the error in offline rendering in the long run, and our result with per-prefix canonical suffixes indeed shows consistently lower relMSE and SMAPE than ReSTIR PT in all scenes.

S.5 PSEUDOCODE

The main document Algorithm 1 includes a high-level pseudocode for our prototype final gather algorithm. In this supplemental document, we provide lower-level pseudocode for selected key components, including GRIS (Algorithm S.3), CRIS (Algorithm S.4), and Gather (Algorithm S.5). We also provide a modified pseudocode for our prototype final gather (Algorithm S.1), listing function parameters more explicitly, and the reservoir structure (Algorithm S.2).

S.6 QUALITY LIMITATIONS

In addition to the cost, our final gather prototype also has some quality limitations. Even with many integration prefixes, our algorithm could have difficulty sampling caustics with sharp directional distributions (due to using BSDF-sampled prefixes). In comparison, ReSTIR PT resamples full paths, allowing easier reproduction of caustics (see the Tower Bridge scene in the interactive image viewer). In addition, slightly increasing the number of borrowed suffixes can sometimes bring considerable noise reduction, especially in scenes with very glossy materials or complex shadows (where the suffix space of the integration prefix is inadequately covered by the suffix space of one nearby prefix). Improving importance sampling of integration prefixes (e.g. by path guiding) and improving neighbor search and reuse efficiency could solve these problems, as mentioned in Section 9.

[1] We use $\mathrm{SMAPE}(I, I_{\mathrm{gt}}) = \mathrm{mean}\left(\frac{\left|I - I_{\mathrm{gt}}\right|}{0.01 \cdot \mathrm{mean}\left(\tilde{I}_{\mathrm{gt}}\right) + (\tilde{I}_{\mathrm{gt}} + \tilde{I})/2}\right)$, where $\tilde{I}$ and $\tilde{I}_{\mathrm{gt}}$ are grayscale input image and ground-truth, respectively.

[2] We use $\mathrm{relMSE}(I, I_{\mathrm{gt}}) = \mathrm{mean}\left(\frac{(I - I_{\mathrm{gt}})^2}{0.01 \cdot \mathrm{mean}\left(\tilde{I}_{\mathrm{gt}}\right)^2 + \tilde{I}_{\mathrm{gt}}^2}\right)$, where $\tilde{I}$ and $\tilde{I}_{\mathrm{gt}}$ are grayscale input image and ground-truth, respectively.

Algorithm S.1: Detailed pseudocode for our prototype final gather. We explicitly show UCW parameters of GRIS and CRIS, but PathContrib and Gather also require these parameters.

```
function SuffixReSTIR()
    parallel foreach pixel q ∈ Image :
        // Temporal supporting prefix update with GRIS.   Rvs: Reservoirs
        q' ← TemporalReprojection(q)
        X^p, W_{X^p} ← TraceNewPrefix(q)   // Canonical sample.
        Rvs[q].(X^p, W_{X^p}) ← GRIS((X^p, W_{X^p}), prevRvs[q'].(X^p, W_{X^p}))   // See Algorithm S.3
        // Temporal suffix update with conditional RIS (CRIS).
        X^s, W_{X^s|Z} ← TraceNewSuffix(Rvs[q].X^p)   // Canonical suffix.
        Rvs[q].(X^s, W_{X^s|Z}) ← CRIS( (X^s, W_{X^s|Z}), prevRvs[q'].(X^s, W_{X^s|Z}), [Rvs[q].X^p, prevRvs[q].X^p])   // See Algorithm S.4
        // Spatial suffix update with CRIS, from random neighbors.
        q_2, ..., q_M ← FindScreenSpaceNeighbors(q)
        Rvs[q].(X^s, W_{X^s|Z}) ← CRIS(Rvs[q].(X^s, W_{X^s|Z}), ..., Rvs[q_M].(X^s, W_{X^s|Z}), [Rvs[q].X^p, ..., Rvs[q_M].X^p])   // See Algorithm S.4
        prevRvs ← Rvs   // Save for next frame.
        // Evaluation (final gather), implements Equation 23.
        for i ← 1 to N do
            X^p, W_{X^p} ← TraceNewPrefix(q)   // Integration prefix.
            [R_1, ..., R_k] ← FindSpatialKNN(Rvs, X^p)   // Find k = M − 1 reservoirs with nearest supporting prefixes.
            // Contributions from reused suffixes.
            Color[q] += Gather([R_1, .., R_k], X^p)/N   // See Algorithm S.5
            if i = 1 then
                // Canonical suffix contribution.
                X^s, W_{X^s|X^p} ← TraceNewSuffix(X^p)
                W_{X^p,X^s} ← W_{X^p} W_{X^s|X^p}
                Color[q] += MIS(X^s, X^p, [X^p, R_1.X^p, ..., R_k.X^p]) · f(X^p, X^s) · W_{X^p,X^s}
            end
        end
    end
```

Technical note:
We must additionally add next event estimation from prefix vertices before the last, and emission until the last, as these contributions are not covered by our suffixes.

Algorithm S.2: The reservoir structure (confidence weights [Lin et al. 2022] are not included for brevity). Note that suffix UCW has an implicit conditional context Z that includes all supporting prefixes (due to spatiotemporal suffix reuse).

```
class Reservoir
    X^p, W_{X^p} ← ∅, 0   // The supporting prefix of X^s
    X^s, W_{X^s|Z} ← ∅, 0   // The suffix, conditional on all supporting prefixes
    w_sum ← 0   // The sum of weights (temporary variable in RIS)
    // Assume whether to update X^p or X^s is clear from the context.
    function update(X_i, w_i)
        w_sum ← w_sum + w_i
        if rand() < (w_i / w_sum) then
            X ← X_i
        end
```

Algorithm S.3: Pseudocode of GRIS for prefix resampling.

```
function GRIS((X_1^p, W_{X_1^p}), (X_2^p, W_{X_2^p}), ··· , (X_M^p, W_{X_M^p}))
    Reservoir r
    for i ← 1 to M do
        Y_i^p ← T_i(X_i^p)   // Shift into the prefix path space Ω^p of pixel 1.
        p̂(Y_i^p) ← f_p(Y_i^p)
        w_i ← m_i(Y_i^p) p̂(Y_i^p) W_{X_i^p} |T_i'(X_i^p)|
        r.update(Y_i^p, w_i)
    end
    if r.X^p ≠ ∅ then
        r.W_{X^p} ← (1 / p̂(r.X^p)) r.w_sum
    end
    return (r.X^p, r.W_{X^p})
```

Algorithm S.4: Pseudocode of CRIS for suffix resampling. Note that $T_i(X_i^s|Z)$ is written as $T_i(X_i^s|\{X_1^p, X_i^p\})$ on line 4 because only X_1^p and X_i^p affect the shift mapping.

1 **function** CRIS($(X_1^s, W_{X_1^s|Z}), \cdots, (X_M^s, W_{X_M^s|Z}), [X_1^p, ..., X_M^p]$)
2 *Reservoir* r
3 **for** $i \leftarrow 1$ **to** M **do**
4 $Y_i^s \leftarrow T_i(X_i^s|\{X_1^p, X_i^p\})$ // Shift into the suffix space $\Omega^s(X_1^p)$.
5 $\hat{p}(Y_i^s) \leftarrow f_{ps}(X_1^p, Y_i^s) f_s(Y_i^s)$
// $m_i(Y^s|Z)$ = MIS(Y^s, current prefix, [all prefixes])
6 $m_i \leftarrow \text{MIS}(Y_i^s, X_i^p, [X_1^p, ..., X_M^p])$
7 $w_i \leftarrow m_i \cdot \hat{p}(Y_i^s) W_{X_i^s|Z} \, |T_i'(X_i^s|\{X_1^p, X_i^p\})|$
8 r.update(Y_i^s, w_i)
9 **end**
10 **if** $r.X \neq \emptyset$ **then**
11 $r.W_{X^s|Z} \leftarrow \frac{1}{\hat{p}(r.X^s)} r.w_{\text{sum}}$
12 **end**
13 **return** $(r.X^s, r.W_{X^s|Z})$

Algorithm S.5: Pseudocode to gather suffix contributions.

1 **function** Gather($[R_1, ..., R_k], X^p, W_{X^p}$)
2 Color $\leftarrow 0$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $Y_i^s \leftarrow T_i(R_i.X^s|\{X^p, R_i.X^p\})$
5 $m_i \leftarrow \text{MIS}(Y_i^s, R_i.X^p, [X^p, R_1.X^p, ..., R_k.X^p])$
6 $W_{Y_i^s|X^p,Z} \leftarrow R_i.W_{X^s|Z} \cdot |T_i'|$
7 $W_{X^p,Y_i^s|Z} \leftarrow W_{X^p} \cdot W_{Y_i^s|X^p,Z}$
8 Color $\leftarrow m_i \cdot f(X^p, Y_i^s) \cdot W_{X^p,Y_i^s|Z}$
9 **end**
10 **return** *Color*

Technical notes:
Z is the supporting prefixes.
Line 4: Given X^p, Z, i'th suffix comes from R_i, chosen by k-NN(X^p; Z).
Line 6: Suffix Y_i^s depends on X^p and Z; condition by current X^p, Z.
Line 7: X^p independent of supporting prefixes Z, so $W_{X^p} = W_{X^p|Z}$.
Line 8: The full estimator is unbiased for any Z.

Table S.1: *Time breakdown for our method with 8 and 32 integration prefixes in Figure 6. "Other" includes components that are not related to the algorithm, like G-Buffer generation and RTXDI.*

	8 prefixes		32 prefixes	
	Time	*%*	*Time*	*%*
Suffix Resampling	**21.69 ms**	**29.3%**	**22.07 ms**	**10.7%**
– Trace New Supp. Prefix	3.50 ms	4.7%	3.65 ms	1.8%
– Supporting Prefix Reuse	1.73 ms	2.3%	1.79 ms	0.9%
– Build temporary BVH	2.59 ms	3.5%	2.61 ms	1.3%
– Trace New Suffix	7.03 ms	9.5%	7.10 ms	3.4%
– Suffix Reuse	6.84 ms	9.2%	6.92 ms	3.4%
Final Gather	**50.42 ms**	**68.1%**	**182.48 ms**	**88.4%**
– Trace Integration Prefixes	21.30 ms	28.8%	72.84 ms	35.3%
– Find Spatial KNN	16.32 ms	22.0%	62.31 ms	30.2%
– Gather Suffix Contribution	12.92 ms	17.3%	47.33 ms	22.9%
Other	**1.92 ms**	**2.6%**	**1.98 ms**	**1.0%**
Total	**74.03 ms**	**100.0%**	**206.53 ms**	**100.00%**

Table S.2: *Timing for Figure 3 (studying gathering from the ReSTIR-driven suffixes (ours) to suffixes sampled by a path tracer (MMIS)).*

Suffix Count	1	2	4	8
MMIS	32 ms	38 ms	54 ms	111 ms
Ours	31 ms	37 ms	58 ms	120 ms

Table S.3: *Timing for Figure 4 that studies the effect of increasing integration prefixes ("128+C" means 128 integration prefixes with a canonical suffix for each).*

Prefix Count	2	8	32	128	128+C
Veach Ajar	25 ms	46 ms	132 ms	482 ms	834 ms
Zero Day	39 ms	71 ms	196 ms	683 ms	1421 ms

Table S.4: *Timings for Figure 1 and 7. All methods use one full path per pixel for integration.*

Scene	Path Tracing	MMIS	ReSTIR PT	Ours
Tower Bridge (Figure 1)	7.0 ms	650 ms	13 ms	750 ms
Veach Ajar (Figure 7)	5.4 ms	426 ms	9.7 ms	489 ms
Zero Day (Figure 7)	12 ms	703 ms	17 ms	717 ms
Classroom (Figure 7)	6.6 ms	385 ms	10 ms	383 ms

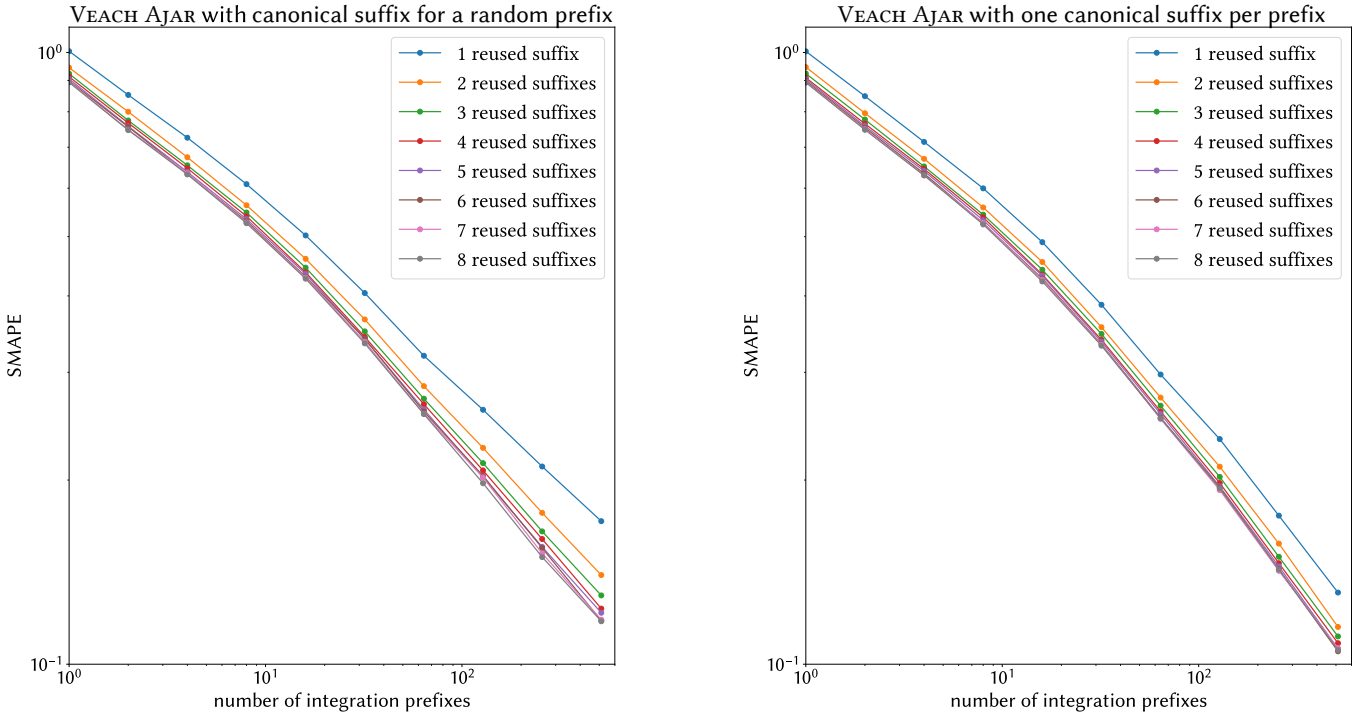

Figure S.3: *SMAPE (symmetric mean absolute percentage error) vs number of integration prefixes in* VEACH AJAR.

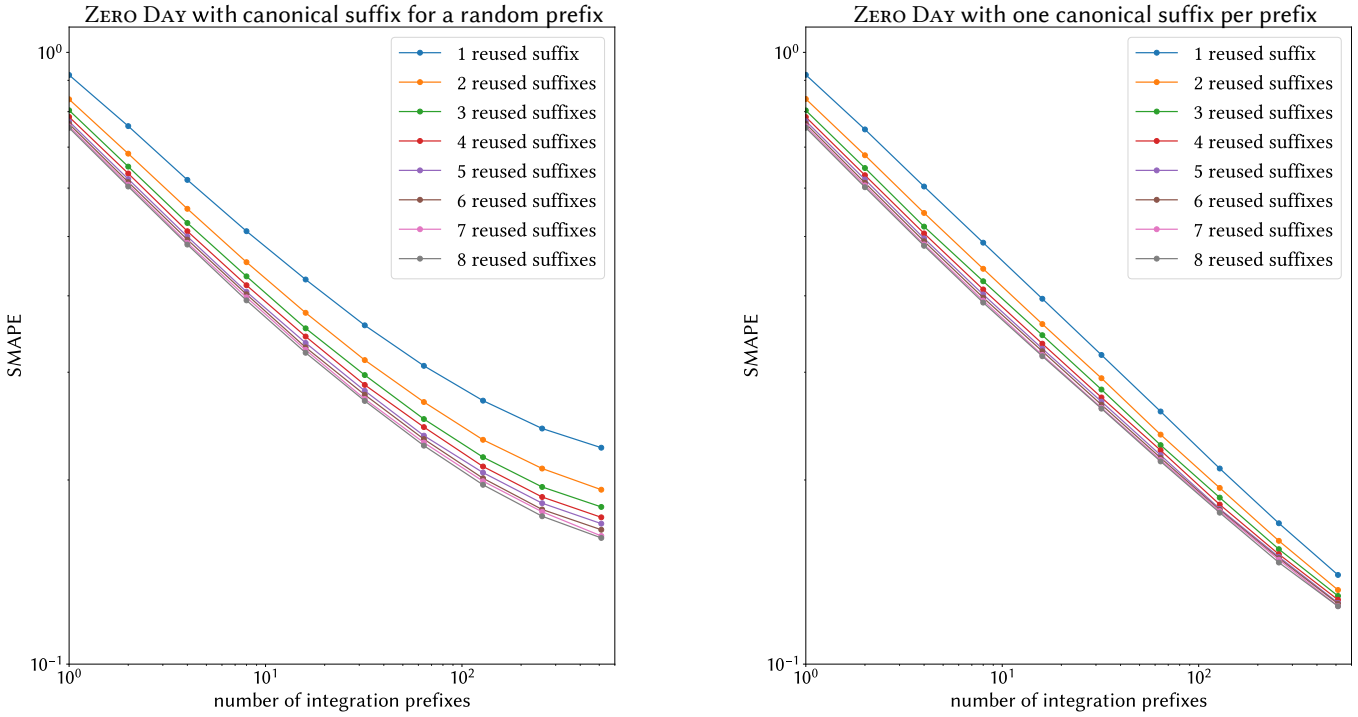

Figure S.4: *SMAPE (symmetric mean absolute percentage error) vs number of integration prefixes in* ZERO DAY.

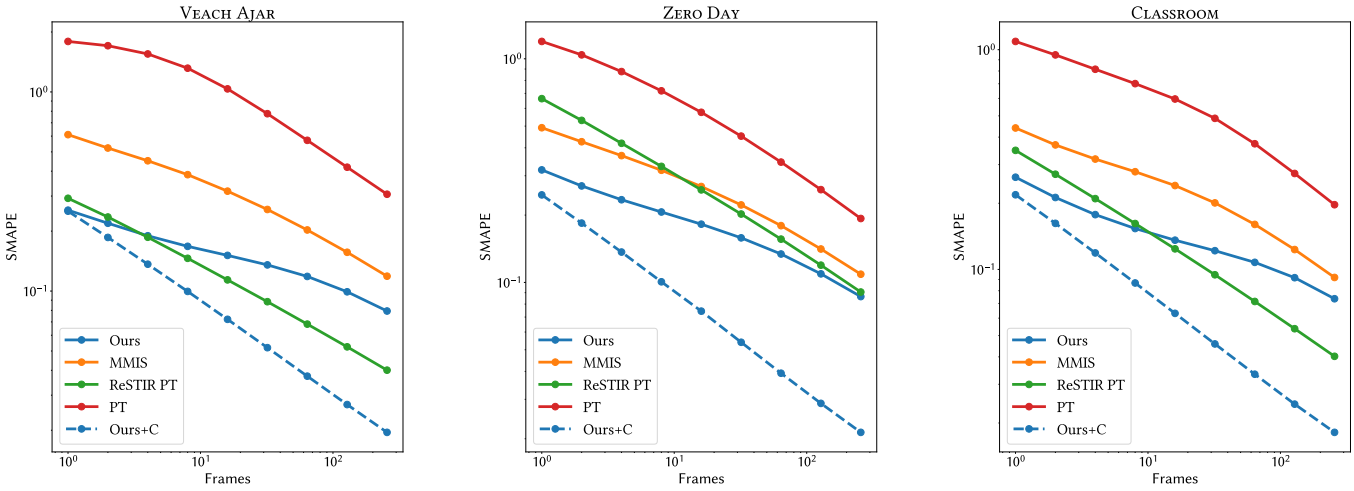

Figure S.5: *Convergence plot for Figure 7 (symmetric mean absolute percentage error), averaging over 1-256 independent renders. Our Russian roulette can sometimes cause increased variance in small parts of the image, which is not reflected in the firefly-resistant SMAPE numbers in Figure 7. Adding per-prefix canonical suffixes ("Ours + C" in the plot) solves the issue.*

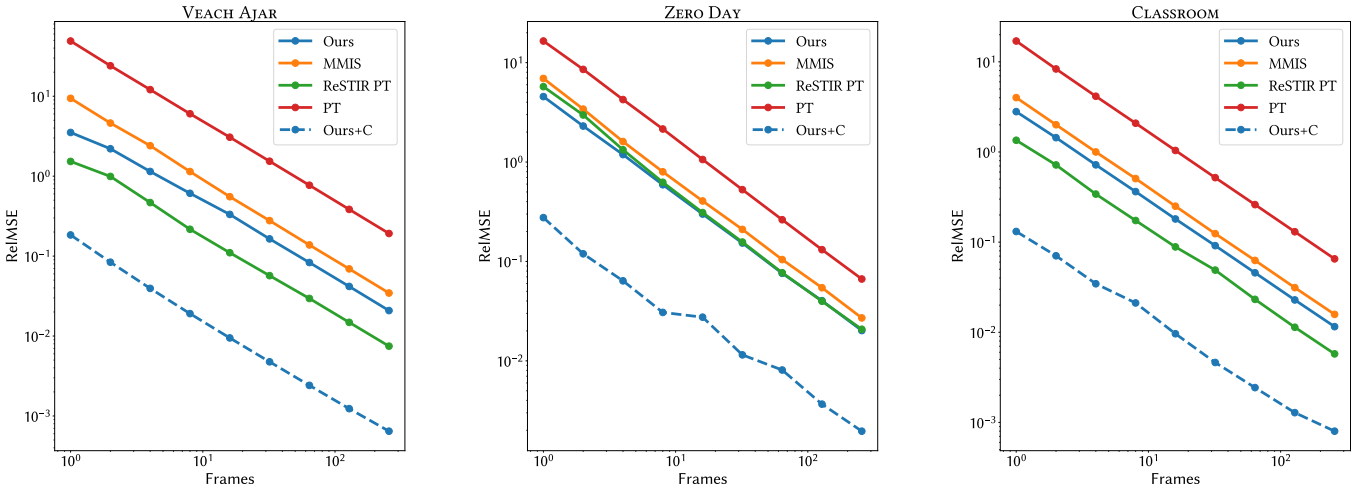

Figure S.6: *Convergence plot for Figure 7 (relative mean squared error), averaging over 1-256 independent renders. Our Russian roulette can sometimes cause increased variance in small parts of the image, which is not reflected in the firefly-resistant SMAPE numbers in Figure 7. Adding per-prefix canonical suffixes ("Ours + C" in the plot) solves the issue. Fluctuation of the curves is mainly caused by direct lighting estimation in RTXDI.*

REFERENCES

Philippe Bekaert, Philipp Slusallek, Ronald Cools, Vlastimil Havran, and Hans-Peter Seidel. 2003. *A custom designed density estimation method for light transport.* Technical Report MPI-I-2003-4-004. Max-Planck-Institut für Informatik. https://domino.mpi-inf.mpg.de/internet/reports.nsf/NumberView/2003-4-004

Benedikt Bitterli. 2021. *Correlations and reuse for fast and accurate physically based light transport.* Ph. D. Dissertation. Dartmouth College. http://benedikt-bitterli.me/Data/dissertation.pdf

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-time Ray Tracing with Dynamic Direct Lighting. *ACM Transactions on Graphics* 39, 4 (2020), 148:1–148:17. https://doi.org/10.1145/3386569.3392481

Iordanis Evangelou, Georgios Papaioannou, Konstantinos Vardis, and Andreas A. Vasilakis. 2021. Fast radius search exploiting ray-tracing frameworks. *Journal of Computer Graphics Techniques* 10, 1 (2021). https://jcgt.org/published/0010/01/02/

Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics* 31, 4 (2012), 1–13. https://doi.org/10.1145/2185520.2185554

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. https://doi.org/10.1145/2766997

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* 41, 4 (2022), 75:1–75:23. https://doi.org/10.1145/3528223.3530158

Matt Pharr, Wenzel Jacob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann. http://www.pbr-book.org/.

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. 419–428. https://doi.org/10.1145/218380.218498