Multi-Layer Reservoir Splatting for Temporal Reuse under Disocclusion

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

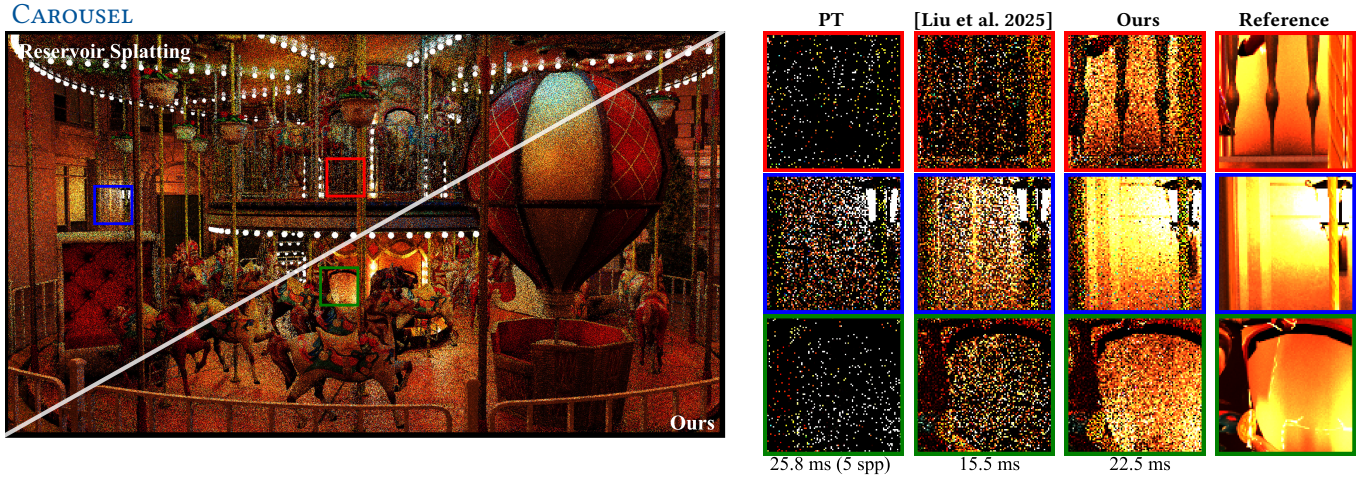

Fig. 1. Comparison of path tracing (PT), reservoir splatting [Liu et al. 2025], and our method on the Carousel scene with many moving objects and frequent disocclusions. Upon geometric occlusion, reservoir splatting entirely discards temporal samples on surfaces that reappear just a few frames later; this manifests as noisy trails around moving objects. In contrast, our method preserves temporal reuse across disocclusions, greatly reducing noise and producing a more consistent noise level across the image.

ReSTIR [Bitterli et al. 2020] efficiently reduces path tracing noise by reusing samples spatiotemporally. Recent ReSTIR methods [Liu et al. 2025] improve temporal reuse from prior frame primary hits. But disocclusions still invalidate some pixel histories, degrading quality with spatially varying noise.

We address disocclusions in ReSTIR by introducing multiple screen-space layers, using reservoir splatting to shift samples between layers. This reuses previously occluded samples that become visible again, reducing noise in disoccluded regions. While shifting samples across layers typically requires tracing multiple rays, we introduce depth ranges and redefine the integration domain to eliminate many ray queries, improving performance. To minimize costs for maintaining many layers, we selectively track only active domains propagated from previous frames.

We validate across multiple scenes, showing greatly reduced disocclusion noise while incurring only a small incremental cost.

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

ACM Reference Format:
Pengpei Hong, Song Zhang, Daqi Lin, Markus Kettunen, Chris Wyman, and Cem Yuksel. 2026. Multi-Layer Reservoir Splatting for Temporal Reuse under Disocclusion. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3799902.3811232

1 Introduction

Recent hardware and algorithmic advances have enabled real-time path tracing for interactive applications. Specifically, reservoir-based spatiotemporal resampling (ReSTIR) [Bitterli et al. 2020; Wyman et al. 2023] significantly improves rendering efficiency at one sample per pixel by reusing path samples spatiotemporally. ReSTIR's quality far exceeds regular path tracing at equal sample count, but its sampling degrades to path tracing when temporal reuse fails, typically when disocclusions make one-to-one pixel mappings between frames impossible. Unfortunately, dynamic scenes exhibit disocclusions regularly, resetting ReSTIR's screen-space sample accumulation and making disocclusions a fundamental challenge for ReSTIR.

Authors' Contact Information: Pengpei Hong, University of Utah, Salt Lake City, USA, pengpei.hong@utah.edu; Song Zhang, University of Utah, Salt Lake City, USA, song.zhang@utah.edu; Daqi Lin, NVIDIA, Redmond, USA, daqil@nvidia.com; Markus Kettunen, NVIDIA, Helsinki, Finland, mkettunen@nvidia.com; Chris Wyman, NVIDIA, Redmond, USA, chris.wyman@acm.org; Cem Yuksel, University of Utah, Salt Lake City, USA, cem@cemyuksel.com.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/26/07
https://doi.org/10.1145/3799902.3811232

Recently, Liu et al. [2025] proposed reprojecting and splatting the previous frame's primary hit samples, guaranteeing each prior sample a chance to contribute and avoiding accidental discards of important sample histories. This better renders high-frequency content. Nevertheless, reprojected samples that become occluded and surfaces invisible in the prior frame can still increase variance.

In this paper, we propose maintaining multiple layers of screen-space reservoirs to avoid discarding samples at disocclusions, allowing recently occluded samples to later become visible without losing their accumulated history. To do this efficiently, we show how to limit the overhead of maintaining multiple layers of reservoirs and define how to shift samples between layers while minimizing the additional ray queries needed to do this in an unbiased way.

Specifically, our contributions include:

- Extending reservoir splatting to handle multiple screen-space layers;
- Efficiently finding which layer to splat samples into, without excessive ray tracing; and
- Masking that keeps multi-layer reservoirs only where they likely contribute to future frames.

We evaluate our method across multiple scenes. Our results demonstrate substantial noise reduction in disoccluded regions while incurring only a small additional computational cost (Figure 1).

2 Related Work

Temporal Reuse. Achieving high fidelity under strict real-time frame budgets has long been challenging. To amortize the expense of shading and sampling, many real-time techniques reuse temporally, accumulating from prior frames to improve the current frame.

Contemporary video games widely use temporal anti-aliasing [Yang et al. 2020] to reduce aliasing via inter-frame color reuse. Similarly, modern upscaling methods such as DLSS [Liu 2020] and FSR [AMD 2022] reconstruct high-resolution images from low-resolution inputs by leveraging temporal reuse. Real-time denoisers [NVIDIA 2020; Schied et al. 2017] also reuse historical samples to reduce Monte Carlo noise, often combined with iterative spatial filtering. All these map current pixels to prior frame pixels via motion vectors, enabling access to the accumulated history.

But a key challenge is determining the reliability of past samples and how to weight their contribution relative to current samples. This is especially difficult under occlusions, geometric changes, or rapid lighting changes. Incorrect decisions often lead to ghosting, flickering, or excessive blurring. Many methods with temporal reuse [AMD 2022; Yang et al. 2020] rely on heuristics based on depth, normal, or color to assess sample validity, often requiring careful tuning for stable quality. Recent learning-based approaches [NVIDIA 2024] predict history sample weights, reducing the reliance on handcrafted heuristics. Despite these advances, disocclusion remains a key problem: newly visible surfaces have no valid temporal data, requiring a fallback to lower-quality spatial filtering, single-frame estimates, or hallucinations. This often leads to visual artifacts.

ReSTIR and Sample Reuse. The ReSTIR family of methods [Wyman et al. 2023] reduces noise in path-traced direct illumination [Bitterli et al. 2020] and global illumination [Kettunen et al. 2023; Lin et al. 2022; Ouyang et al. 2021] by applying resampled importance sampling (RIS) [Talbot et al. 2005] spatiotemporally to improve sample distributions. ReSTIR inherently weights history samples correctly through its target function, shift mappings, and confidence weights, allowing principled temporal reuse without introducing bias.

But ReSTIR still relies on temporal reuse for high-quality results at low sample counts. When temporal reuse fails, sample quality largely degrades to that of path tracing. Recent work thus aims to maintain temporal reuse whenever possible [Liu et al. 2025; Zhang et al. 2024]. Unfortunately, these methods do not address disocclusions, so these regions continue to exhibit much more noise.

Deep Frame Buffers. Our solution to disocclusion involves storing per-pixel data that can contain multiple layers, analogous to deep frame buffers. In the context of rasterization, deep frame buffers are often generated via *depth peeling* [Everitt 2001], originally targeting order-independent transparency by separately rendering multiple depth layers. Subsequent work [Bavoil and Myers 2008; Liu et al. 2009b,a] improved performance within the rasterization pipeline.

Many use cases [Vasilakis et al. 2020] for depth peeling have been proposed. For example, Mara et al. [2016] store multiple depth layers to improve the stability of raster-based, screen-space global illumination approximations. Our method also peels multiple surfaces along a view ray, but it serves a different purpose: enabling temporal unbiased sample reuse around depth discontinuities.

3 Preliminaries

ReSTIR builds on generalized resampled importance sampling (GRIS) [Lin et al. 2022]. For a given sampling problem performed over different domains (domains corresponding to different pixels and samples corresponding to paths), GRIS draws candidate samples X_i from multiple source domains Ω_i and maps them to a common target domain Ω using bijective shift mappings $T_i : \Omega_i \to \Omega$. The goal of GRIS is resampling from these candidates such that the resulting sample distribution approaches some target function $\hat{p}$.

For each candidate, we evaluate the shifted sample $Y_i = T_i(X_i)$, the target function value $\hat{p}(Y_i)$, the shift mapping's Jacobian $|\partial T_i / \partial X_i|$, and the unbiased contribution weight (UCW) W_{X_i}. The resampling weights are then given by

$$w_i = m_i(Y_i)\, \hat{p}(Y_i)\, W_{X_i} \left| \frac{\partial T_i}{\partial X_i} \right| , \tag{1}$$

where $m_i(y)$ are MIS weights defined over the domain Ω and satisfy $\sum_{i=1}^{M} m_i(y) = 1$ with $m_i(y) = 0$ for Ω_i that cannot produce y.

A sample Y is selected from the choices Y_i with probability proportional to w_i, and its UCW is computed as

$$W_Y = \frac{1}{\hat{p}(Y)} \sum_{i=1}^{M} w_i \, . \tag{2}$$

As $\mathrm{Var}[\sum_i w_i] \to 0$, the distribution of Y converges to $\hat{p}$, and the estimator $f(Y)\, W_Y$ yields a low-variance estimate of $\int_\Omega f(y)\, \mathrm{d}y$ when $f \approx \hat{p}$.

ReSTIR applies GRIS independently per pixel to reuse samples spatiotemporally. Each pixel maintains a reservoir. Initially populated by a naive path tracer, this reservoir is then resampled by sourcing candidates from neighboring or prior frame reservoirs.

Earlier ReSTIR methods estimated point-sampled integrals of the radiance received through the pixel centers. Area ReSTIR [Zhang et al. 2024] extends the integration domain to the full pixel footprint, directly estimating each pixel's filtered color and enabling anti-aliasing across frames with a single new sample per pixel.

To perform temporal reuse at a pixel, Area ReSTIR backprojects a pixel into the prior frame (at an off-grid 1×1 region) by following fractional motion vectors, and resamples prior samples within this off-grid region. This makes temporal reuse much more robust for subpixel, high-frequency normal maps. However, it comes at a cost and fails when subpixel details have multiple motion vectors, like at disocclusions.

Reservoir splatting [Liu et al. 2025] improves over Area ReSTIR, by splatting prior-frame samples directly into the current frame, connecting their shifted primary hits (usually moved with the objects) to the camera. Each splatted sample directly contributes to the pixel it lands on. Compared to Area ReSTIR, this approach increases temporal sample utilization while replacing the expensive ray tracing of new primary hits with cheaper visibility queries.

4 Multi-Layer Reservoir Splatting

ReSTIR forwards accumulated reservoir information to the next frame via temporal reprojection. But screen-space projection breaks under occlusion; when a surface becomes occluded, its reservoir state cannot be carried forward. If occluded surfaces later reappear, they return without any temporal history, restarting with a low-sample noisy estimate. This often appears as a trail of noise following moving objects, and can easily be misperceived as a ghostlike "shadow" post-denoising.

Our method overcomes this problem by storing multiple reservoir layers, akin to a deep frame buffer [Vasilakis et al. 2020]. Our method builds on reservoir splatting. In a single-layer setup, a splatted sample gets discarded if it is occluded and fails a visibility test. With multiple layers, occluded samples can be assigned to a reservoir in a deeper layer. This allows occluded samples to contribute if they later become visible.

To avoid storing all occluded samples, we adaptively allocate deep layers. Specifically, we mark domains (i.e., a layer in a pixel) that are likely to reappear soon due to disocclusions as *active* and forward them to the next frame (Section 4.3).

In the following, we first formulate our multi-layer reservoirs (Section 4.1), then introduce efficiency techniques: a depth-range formulation (Section 4.2) that reduces the ray queries needed to choose the splat layer, and selective maintenance of deep-layer information using active domains (Section 4.3).

4.1 Multi-layer Reservoir Formulation and Splatting

Instead of maintaining reservoirs only for primary hits, we introduce additional buffers that store per-pixel second-nearest hits, third-nearest hits, and so on. In a ray tracer, additional layers are relatively cheap to generate by obtaining additional intersections beyond the first hit.

To extend the path integral[1] for multi-layer resampling, consider the original path integral $I_1 = \int_\Omega f(\bar{\mathbf{x}}) \mathrm{d}\bar{\mathbf{x}}$ estimated by the first layer

[1]Our path integral notation drops the pixel index and pixel filter terms for simplicity.

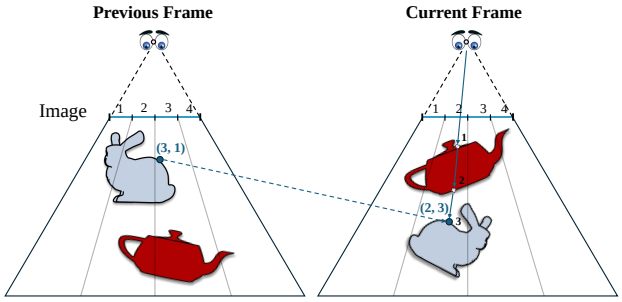

Fig. 2. Visualization of a temporal shift from a prior domain to the current domain. Here a domain is a pixel-layer pair (p, i). In the previous frame, the sample of $(3, 1)$, i.e., the first layer of pixel 3, lies on the bunny, which is visible to the camera. Due to scene changes, it moves with the bunny and projects to pixel 2 in the current frame. To determine its layer, we trace a ray from the camera toward the reprojected sample. After each intersection, we spawn a continuation ray in the same direction. The number of intersections before reaching the sample defines the layer index. In this example, the resulting layer index is 3, so the sample belongs to the domain $(2, 3)$.

of a pixel, where Ω is the path space. Here, a path $\bar{\mathbf{x}}$ with vertices $\bar{\mathbf{x}} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ...]$ has a chance to contribute only if its visibility $V(\mathbf{x}_0, \mathbf{x}_1) = 1$, i.e. $\mathbf{x}_1$ is visible from $\mathbf{x}_0$. Assuming a pinhole camera , for the i-th screen-space layer, we define the set of i-th nearest hits

$$\mathcal{A}_i = \{\, x \mid \mathrm{rank}(x) = i \,\},$$

where $\mathrm{rank}(x)$ denotes the order of intersection of x along the camera ray. $\mathrm{rank}(x) = 1$ if x is the first hit along the ray (directly visible), and it increments by one for every additional intersection required to reach x. The i-th layer integral computes the radiance under the assumption that the rank-i intersections are directly visible from the camera, i.e.

$$I_i = \int_\Omega [\mathbf{x}_1 \in \mathcal{A}_i] g(\bar{\mathbf{x}}) \mathrm{d}\bar{\mathbf{x}}\,, \tag{3}$$

where $g(\bar{\mathbf{x}})$ is $f(\bar{\mathbf{x}})$ with the $V(\mathbf{x}_0, \mathbf{x}_1)$ term removed. Identified by the pixel-layer indices (p, i), each reservoir therefore integrates over the pixel-layer domain defined by paths whose first surface vertex has the same rank i and projects to pixel p.

This multi-layer formulation enables temporal reuse of samples that were previously occluded: when a pixel becomes disoccluded, its current front layer integral I_1 often matches some deeper-layer estimate I_i^{prev} from the previous frame because the valid set of $\mathbf{x}_1$ coincides or closely matches. Any difference between integrals grows only insofar as the occluder motion changes the remaining path space (e.g., the occluder blocking a light source reaching areas behind it), in which case a lighting condition change has happened and temporal reuse is inherently unreliable. The same reasoning applies to reuse between deep layers, allowing us to preserve sampling information under higher depth complexity.

Reservoir splatting provides a convenient mechanism for reusing temporal samples across layers. During shift mapping, a temporal sample is splatted to a specific pixel and particular layer. Assuming N pixels with k layers, conceptually, we splat each of kN domains to kN domains, but only $O(kN)$ operations are needed [Liu et al. 2025].

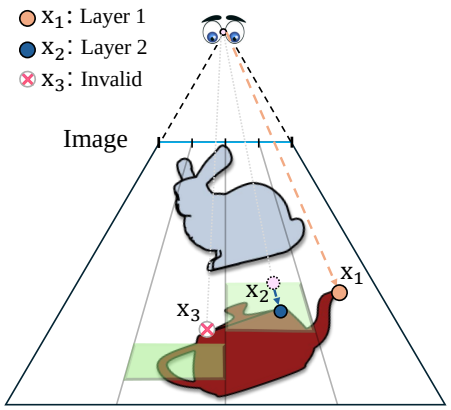

Fig. 3. Illustration of using depth ranges to determine the target layer of a shifted sample. Depth ranges are shown in green. Sample $\mathbf{x}_1$ is directly visible to the camera and is therefore assigned to the front layer. Sample $\mathbf{x}_2$ is occluded but falls within the depth range of the second layer; a visibility ray (shown as a blue arrow) restricted to this range confirms that it is the closest hit, and it is assigned to layer 2. Sample $\mathbf{x}_3$ is occluded and does not fall within any depth range, and is therefore considered invalid in the current frame.

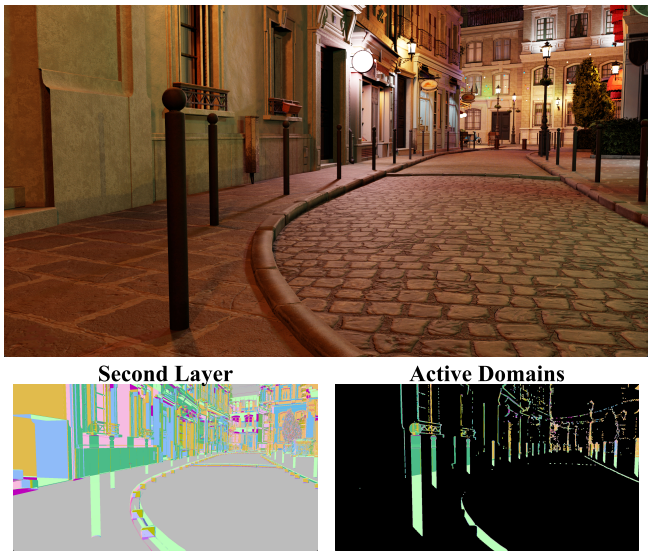

Fig. 4. Visualization of all domains and active domains in the second layer in the Bistro scene. Surface normals of the center hits are shown in false color (gray indicates no surface hit), and inactive domains are marked in black. Our method preserves only surfaces that were visible in previous frames, resulting in a very small maintained subset of the full screen.

To determine the target layer, we trace rays along the camera direction toward the splatted sample and count surface intersections until the corresponding hit is reached. Specifically, we iteratively trace camera rays whose origins are advanced to lie just beyond previously detected intersections, thereby enumerating successive surface hits along the same ray. The number of intersections encountered before reaching the sample determines its layer index. Figure 2 illustrates this process.

In the rest of this section, we describe how to make the multi-layer splatting method efficient in practice.

4.2 Depth Ranges

As described above, shift mapping can be expensive; determining the target layer of a splatted sample may require multiple rays to step through all occluders. We reduce costs by introducing *depth ranges*, allowing us to replace multiple closest-hit rays with two visibility queries.

We observe that strictly enforcing the original layer definition $\mathcal{A}_i = \{ x \mid \text{rank}(x) = i \}$ is unnecessary for deeper layers, as the final image is covered by the front layer $\mathcal{A}_1$. For deeper layers, we relax this requirement by precomputing occluder depth ranges and assigning samples to layers by camera distance. This coarser partition may discard some samples (only the closest surface in each range is treated as valid), providing less variance improvement, but it substantially lowers the cost of layer assignment.

Concretely, we define non-overlapping depth ranges $D_i = [L_i, R_i]$ for each layer, and redefine the deeper domains ($i > 1$) as

$$\mathcal{A}_i = \{ x \mid d(x, \mathbf{x}_0) \in D_i,\ V(x, \mathbf{x}_0) = 0,\ V(x, \mathbf{x}_0 + L_i \mathbf{r}(x)) = 1 \}, \quad (4)$$

where $d(x, \mathbf{x}_0)$ is the camera distance of x and $\mathbf{r}(x)$ is the unit vector pointing from $\mathbf{x}_0$ to x. These conditions ensure that x is not directly visible from the camera, yet becomes the first visible intersection starting at the near bound L_i of its depth range.

To better approximate the original layer assignment, we allow each pixel to use a different set of depth ranges. Specifically, we use the intersections along the ray shot from the pixel center as *representative hits* for that pixel, denoted as $\mathbf{c}_{p,i}$ for layer i in pixel p. The depths of these representative hits are expanded to non-overlapping intervals to form depth ranges, defining the layers of that pixel. This is only precomputed once at the start of each frame.

With non-overlapping ranges, layer selection for a splatted sample becomes efficient. For the hit point x associated with a splatted sample, we first trace a camera visibility ray to test whether x belongs to $\mathcal{A}_1$. If it does not, we identify the depth range D_i that contains $d(x, \mathbf{x}_0)$ and trace a visibility ray restricted within D_i to test whether $\mathbf{y}$ is the closest hit within that range. If so, the sample is assigned to the i-th layer.

The process of determining the layer is shown in Figure 3 and further implementation details are provided in Section 5.

4.3 Active domains

Maintaining k layers for N pixels results in kN pixel-layer domains, increasing the computational cost of ReSTIR by a factor of k. In practice, however, many distant objects have a low probability of becoming disoccluded for most pixels. For example, in indoor scenes, samples originating from distant rooms are unlikely to become visible and therefore do not need to be maintained.

Therefore, we maintain only a subset of *active* domains that are likely to reappear. Inactive domains do not need to generate new path samples and participate in splatting. The main challenge is

deciding which pixel-layer domains are active. Our key observation is that if a sample was a primary hit in previous frames and subsequently becomes occluded, it is likely to become visible again in the future and should therefore be preserved.

This leads to a transitive activation scheme. In the first frame, only primary-layer domains are active. When a previous pixel-layer domain (p, i) is marked as active, we splat its representative hit $\mathbf{c}_{p,i}$ into the current frame and mark the corresponding domain (p', i') as active. The newly visible domains are also marked as active. Since the domains are stored in screen space, a domain automatically becomes inactive if its representative hit splats out of the screen. Since splatting can introduce holes in screen space [Liu et al. 2025], we additionally improve spatial continuity using a simple hole-filling strategy: we treat neighboring domains within a fixed radius of the current active domains as active. These neighboring domains participate in sampling, but do not activate the next frame's domains. In our experiments, this simple scheme effectively preserves local context and yields stable resampling results. Figure 4 visualizes the active domains maintained by our method.

With depth ranges and selectively activated domains, our method achieves visual quality comparable to the original multi-layer reservoir splatting, while significantly reducing computational cost by reducing the number of maintained domains and ray queries.

5 Implementation Details

We perform a walkthrough of our algorithm in this section that contains key implementation details. A reservoir represents a pixel-layer domain. Each domain stores two surface hits: a representative hit (traced from pixel center) that represents the domain and a sub-pixel sample hit used to estimate the pixel integral.

At each frame, we first propagate domains by splatting the representative hits of active domains from the previous frame into the current frame. The newly activated layer index i' is determined by counting the number of intersections from the shifted representative hit $T(\mathbf{c}_{p,i})$ to the new camera (T transforms $\mathbf{c}_{p,i}$ by preserving its object-space location). Domain hole-filling around these propagated active domains is applied as described before, and we use a filling radius of 3 pixels. We obtain the new representative hits of the activated domains $\mathbf{c}_{p',i'}$ by tracing rays from the pixel centers.

Since we propagate active domains using representative hits which are point samples, there can be approximation errors that lead to false positives – domains that are unwantedly activated. To reduce the false positives that penalize performance, we additionally compare the immediate occluders of $T(\mathbf{c}_{p,i})$ [2] and $\mathbf{c}_{p',i'}$. If the two occluders are different, the domain is not activated.

Then, we compute the depth ranges for the activated domains. The depth range of each domain is derived from the depth t of its representative hit. In our implementation, we define the depth range as $[(1-\epsilon)t,\ (1+\epsilon)t]$ and use $\epsilon = 0.01$, which we found to work well across all tested scenes. To ensure that depth ranges do not overlap for a given pixel, if two adjacent ranges $[L_i, R_i]$ and $[L_{i+1}, R_{i+1}]$ overlap, we clamp their boundaries by setting

$$R'_i = L'_{i+1} = \frac{R_i + L_{i+1}}{2}.$$

[2] In our implementation, occluders are compared using instance IDs.

The steps above initialize the domains and their attributes for the current frame's ReSTIR. In the initial resampling phase, we generate canonical samples for each active domain in the current frame. This is important for maintaining an unbiased estimate of the corresponding integral.

In temporal resampling, we perform multi-layer splatting. We first splat the samples of all active domains in the prior frame to the current frame. Note that samples that do not fall within the depth range of any active domain are rejected. As in Liu et al. [2025], we also need to splat the newly generated samples in all current frame's active domains to the previous frame to compute MIS weights.

To preserve unbiasedness during temporal reuse, we update the confidence weight of each active domain by projecting its representative hit back to the previous frame and reusing the confidence weight stored in the corresponding reservoir, similar to Reservoir Splatting [Liu et al. 2025].

We only perform spatial reuse for the front layer $\mathcal{A}_1$ as we found that the quality improvement of performing spatial reuse for deeper layers does not justify the additional cost.

As a storage optimization, we compact all active domains into a contiguous buffer and assign one thread per domain to improve thread utilization. This is based on the observation that deep domains are often sparsely distributed across the image.

Finally, since back-facing surfaces do not contribute to the rendering results, we exclude them from domain construction. In practice, this corresponds to retaining only odd-numbered surface intersections, where layer i corresponds to the $(2i-1)$-th surface intersection along the camera ray.

6 Results

We implemented our method in Falcor [Kallweit et al. 2022] and evaluated it on a PC equipped with an Intel i9-13900K CPU and an NVIDIA RTX 4090 GPU. Error values are reported using MAPE and FLIP [Andersson et al. 2020] [3]. All images are rendered at the resolution of 1920×1080. We evaluate our method on the Bistro, Subway, Emerald Square, and Carousel scenes. Camera motion is applied in Bistro, Subway, and Emerald Square, while object motion is present in Carousel. We set the maximum layer count based on (visible) scene depth complexity, which ranges from 2 to 5 in our results.

Equal-time Comparisons. We compare our method against Reservoir Splatting [Liu et al. 2025] and two of its variants. Since Reservoir Splatting is not designed to handle disocclusions, we extend it in two straightforward ways for disoccluded pixels only: increasing the number of samples per pixel (SPP) during the initial RIS stage ("more initial spp"), and increasing the number of spatial neighbors for spatial reuse ("more neighbors"). Both parameters are adjusted to match the overhead of our method.

As shown in Figure 9, Reservoir Splatting exhibits high variance in disoccluded regions, whereas our method substantially reduces variance. By reusing high-quality temporal samples stored in occluded layers, our approach maintains stable cost and quality regardless of disocclusion size or initial sampling difficulty. In contrast, Reservoir

[3] Our method yields diminishing quantitative gains because errors are computed over the entire image, whereas disoccluded regions occupy only a small portion of the frame.

Table 1. Statistics and timings for Reservoir Splatting ("Base") and our method in Figure 9. We report the number of maintained domains for the non-adaptive approach ("Uniform") and our selective activation method using the same number of layers. We also report the number of ray traces and splatting times with and without depth ranges.

	Timings (ms)				*Domain Effectiveness*				*Effect of Depth Ranges (Splatting)*		*Memory*
Scene	Domain Init (Ours)	Initial Resampling (Base / Ours)	Temporal Splatting (Base / Ours)	Spatial Resampling	Disoccluded Pixels	Deep Domains (Uniform / Ours)	Disoccluded Pixels Covered (Uniform / Ours)	Coverage Rate (Uniform / Ours)	Rays Traced (w/o / w/ ranges)	Splatting Time (ms) (w/o / w/ ranges)	Memory Usage (Base / Ours)
Bistro	0.77	1.56 / 1.80	1.58 / 1.89	2.24	27618	2.07M / 0.26M	24306 / 20717	88.00% / 75.01%	8.22M / 7.56M	2.13 / 1.89	380MB / 428MB
Emerald Square	1.22	2.37 / 3.2	2.17 / 3.05	2.86	294318	3.23M / 0.79M	223010 / 194452	73.63% / 66.07%	17.42M / 9.29M	4.6 / 3.05	380MB / 525MB
Subway	1.27	0.57 / 1.17	0.86 / 1.70	1.42	352182	5.74M / 1.21M	345006 / 317385	97.96% / 90.11%	17.00M / 10.70M	2.14 / 1.70	380MB / 602MB
Carousel	1.97	2.47 / 4.55	2.13 / 3.93	4.12	247573	7.32M / 1.91M	240164 / 218358	97.00% / 88.20%	29.32M / 14.41M	6.05 / 3.93	380MB / 731MB

Splatting must increase SPP or the spatial neighbor count to adapt to these factors, yet still falls short of our quality.

Comparison against SPMIS. We compare our method with Stochastic Pairwise MIS (SPMIS) [Hedstrom et al. 2026] in Figure 10. SPMIS mitigates disocclusion noise by reusing neighboring pixels with contributing samples, rather than randomly selecting neighbors. This approach performs well when high-quality samples are available for spatial reuse, but becomes less effective when neighboring samples are uninformative (e.g., under complex lighting or large spatial variation), and it does not ensure temporal stability in the presence of subpixel geometries such as foliage.

Instead, our method focuses on improving temporal reuse and preserving sample quality with accumulated history. However, it does not address noise arising from newly visible geometry absent in previous frames (e.g., noise near screen boundaries). Combining our method with SPMIS leverages the strengths of both approaches and achieves the best overall quality.

Ablation on the number of layers. Figure 11 illustrates the effect of varying the number of screen-space layers on variance reduction in disoccluded regions. Increasing the number of layers progressively improves the variance for different disoccluded regions, as more layers allow storing more high-quality deep samples. Scenes with more overlapping geometry, therefore, require more layers to effectively recover disoccluded samples. Yet, the cost increase is relatively minor due to maintaining only active domains.

Validations. Figure 5 compares Reservoir Splatting, a uniform five-layer implementation without depth ranges, and our optimized approach on the Carousel scene. Compared to Reservoir Splatting, our method reduces noise in most disoccluded regions. While our selective activation strategy does not cover every disocclusion case, it achieves an overall noise level comparable to the uniform five-layer implementation at a much lower cost. This demonstrates that selectively tracking only domains that were visible in previous frames is sufficient to handle most disocclusions in practice. However, when not all primary domains are preserved in the previous frame, some residual noise can appear (yellow inset).

Figure 6 compares our method with and without hole filling for domain activation. Holes can arise naturally during the splatting of representative hits, leaving some domains inactive. When such inactive domains later become disoccluded, the absence of temporal samples leads to increased noise. Hole-filling mitigates this issue by improving spatial continuity in domain activation, thereby reducing disocclusion noise. We also observe that increasing the filling radius yields diminishing returns under typical animations or camera motion.

Statistics Analysis. Table 1 presents a detailed performance breakdown for each stage. Compared to the Reservoir Splatting baseline, our method adds a domain initialization stage and introduces overhead to initial resampling and temporal resampling. We further evaluate the effectiveness of our domain activation heuristic by measuring how many disoccluded pixels are covered, and comparing our method against the uniform k-layer implementation. Despite maintaining significantly fewer domains, our method covers most disoccluded pixels and achieves coverage rates comparable to the uniform approach. This demonstrates that our method effectively focuses computation on important domains while avoiding unnecessary maintenance overhead. Table 1 also shows that introducing depth ranges reduces the number of rays traced during splatting by up to 59%. We also report estimated memory usage, which scales with the number of maintained domains.

NRD results. Denoising is essential for real-time path tracing, yet disocclusion remains challenging for most temporal denoisers. We evaluate how our method affects denoised results in disoccluded regions using NVIDIA Real-Time Denoisers (NRD) [NVIDIA 2020]. Figure 7 compares Reservoir Splatting and our method under NRD on the Bistro scene. In disoccluded foliage regions, Reservoir Splatting produces noticeable color blobs due to high input variance. By reusing higher-quality history samples from occluded layers, our method provides a less noisy input signal, enabling NRD to produce a more temporally consistent image without introducing additional spatial artifacts.

NRD also provides a disocclusion-handling option (referred to as the *disocclusion fix*). We evaluate its behavior with both Reservoir Splatting and our method in Figure 8. This feature detects disoccluded regions that lack reliable temporal denoising history and applies aggressive spatial filtering, which can lead to excessive blurring. This behavior is observed even when using our method, highlighting a fundamental limitation of temporal denoising under severe disocclusion. This shows that our method provides a better solution than NRD's disocclusion handling feature.

7 Limitations

Our method operates entirely in screen space and stores only screen-space information. As a result, it does not reduce artifacts caused

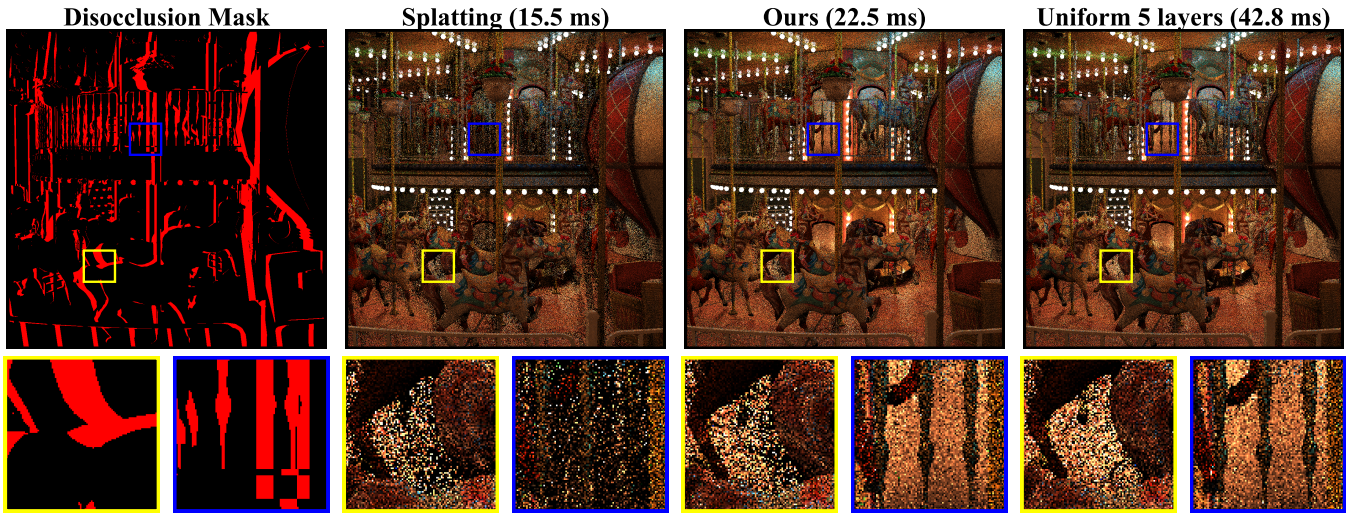

Fig. 5. Comparison between Reservoir Splatting, a uniform five-layer implementation, and our method on the Carousel scene. A disocclusion mask is shown, where red pixels indicate disoccluded regions in the current frame. Note that noisier regions are not confined simply to the current frame's disocclusion regions, due to time required to recover from prior-frame disocclusions. Compared to Reservoir Splatting, our method reduces noise in disoccluded areas, while achieving an overall noise level comparable to the uniform five-layer implementation at much lower cost.

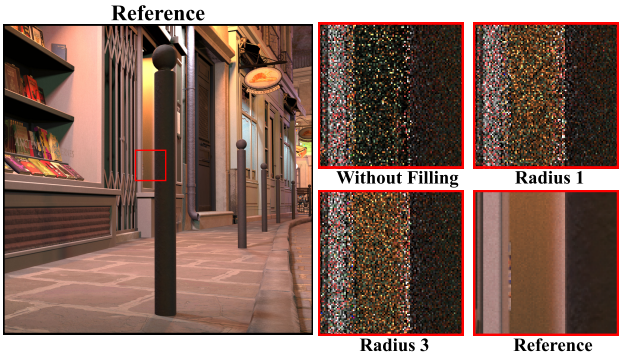

Fig. 6. Example showing how hole-filling avoids introducing extra noise when propagating active domains under camera motion. Increasing the filling radius brings diminishing improvements.

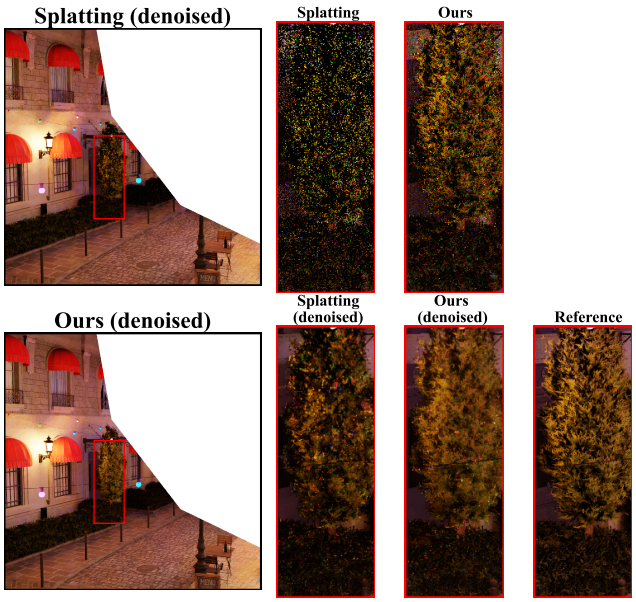

Fig. 7. Comparison of Reservoir Splatting and our method with NRD denoising applied. A moving white light bulb near the camera creates a large disocclusion region. By preserving higher-quality temporal samples in disoccluded foliage regions, our method reduces the variance of denoiser inputs and enables NRD to produce results that better preserve details and more closely match the reference.

by geometry entering the view frustum, such as the noisy screen boundaries in Figure 9. Extending our method to address such cases would likely require maintaining out-of-screen samples, which we do not consider in this work.

Our selective domain activation approach trades coverage for reduced overhead. We predict disocclusion based on samples that were visible in previous frames. Consequently, regions that become visible for the first time may initially exhibit higher noise.

When a large number of previously visible samples become occluded, our method may need to maintain many domains because those samples could reappear later. In such cases, employing a strategy that avoids updating deep reservoirs every frame could significantly lower the computational cost.

Our current formulation does not support depth of field or motion blur. In these settings, determining whether a sample is occluded or disoccluded becomes non-trivial, as visibility must be evaluated over the entire lens aperture or across the full temporal integration interval.

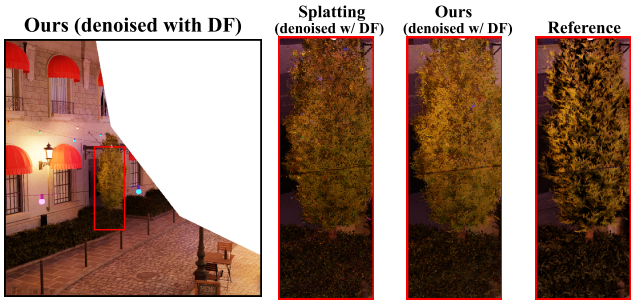

Fig. 8. NRD results with its disocclusion fix ("DF") feature enabled. The disoccluded regions are heavily blurred regardless of the input quality. It reduces the fireflies of the baseline but overblurs our result.

8 Conclusion and Future Work

We introduce multi-layer reservoir splatting, which enables samples to be splatted across screen-space layers and allows high-quality occluded samples to contribute once they become visible. Using depth ranges, splatted samples can determine their target layer with two visibility queries rather than multiple ray traces. To limit overhead, we do not maintain all pixel–layer pairs; instead, we selectively track only active domains propagated from previous frames. Our method substantially reduces the disocclusion noise of ReSTIR in realistic rendering scenarios at modest additional cost.

An interesting direction for future work is to expose the occluded sample information maintained by our method to temporal techniques such as upscalers and denoisers to improve handling of disoccluded regions. Another promising direction is to extend our approach to transparency rendering, e.g., by tracking opaque samples behind transparent surfaces so their information can be reused as objects move around.

References

AMD. 2022. FidelityFX Super Resolution 2.0. https://gpuopen.com/fidelityfx-superresolution-2/. Accessed: 2026-01-20.

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 15 (Aug. 2020), 23 pages. doi:10.1145/3406183

Louis Bavoil and Kevin Myers. 2008. Order independent transparency with dual depth peeling. *NVIDIA OpenGL SDK* 1, 12 (2008), 2–4.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 148 (Aug. 2020), 17 pages. doi:10.1145/3386569.3392481

Cass W. Everitt. 2001. Interactive Order-Independent Transparency. https://api.semanticscholar.org/CorpusID:5813703

Trevor Hedstrom, Markus Kettunen, Daqi Lin, Chris Wyman, and Tzu-Mao Li. 2026. Stochastic Pairwise MIS for Unbiased Large-Kernel Reuse in Real Time. 45, 2 (May 2026).

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš 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 https://github.com/NVIDIAGameWorks/Falcor.

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

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, 4, Article 75 (July 2022), 23 pages. doi:10.1145/3528223.3530158

Baoquan Liu, Li-Yi Wei, Ying-Qing Xu, and Enhua Wu. 2009b. Multi-layer depth peeling via fragment sort. In *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics*. IEEE, 452–456.

Edward Liu. 2020. DLSS 2.0 – Image Reconstruction for Real-Time Rendering with Deep Learning. NVIDIA Technical Presentation. http://behindthepixels.io/assets/files/DLSS2.0.pdf

Fang Liu, Meng-Cheng Huang, Xue-Hui Liu, and En-Hua Wu. 2009a. Efficient depth peeling via bucket sort. In *Proceedings of the Conference on High Performance Graphics 2009*. 51–57.

Jeffrey Liu, Daqi Lin, Markus Kettunen, Chris Wyman, and Ravi Ramamoorthi. 2025. Reservoir Splatting for Temporal Path Resampling and Motion Blur. *SIGGRAPH (Conference Track)*. doi:10.1145/3721238.3730646

Michael Mara, Morgan McGuire, Derek Nowrouzezahrai, and David Luebke. 2016. Deep G-Buffers for Stable Global Illumination Approximation. In *Eurographics/ACM SIGGRAPH Symposium on High Performance Graphics*, Ulf Assarsson and Warren Hunt (Eds.). The Eurographics Association. doi:10.2312/hpg.20161195

NVIDIA. 2020. NVIDIA Real-Time Denoisers (NRD) SDK. Available at https://github.com/NVIDIA-RTX/NRD. [Online; accessed 20-January-2026].

NVIDIA. 2024. Rays Up: Decoding AI-Powered DLSS 3.5 Ray Reconstruction. https://blogs.nvidia.com/blog/ai-decoded-ray-reconstruction/. Accessed: 2026-01-22.

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

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (Los Angeles, California) *(HPG '17)*. Association for Computing Machinery, New York, NY, USA, Article 2, 12 pages. doi:10.1145/3105762.3105770

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering (2005)*, Kavita Bala and Philip Dutre (Eds.). The Eurographics Association. doi:10.2312/EGWR/EGSR05/139-146

A. A. Vasilakis, K. Vardis, and G. Papaioannou. 2020. A Survey of Multifragment Rendering. *Computer Graphics Forum* 39, 2 (2020), 623–642. doi:10.1111/cgf.14019

Chris Wyman, Markus Kettunen, Daqi Lin, Benedikt Bitterli, Cem Yuksel, Wojciech Jarosz, and Pawel Kozlowski. 2023. A Gentle Introduction to ReSTIR Path Reuse in Real-Time. In *ACM SIGGRAPH 2023 Courses* (Los Angeles, California) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 1, 38 pages. doi:10.1145/3587423.3595511

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

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. doi:10.1145/3658210

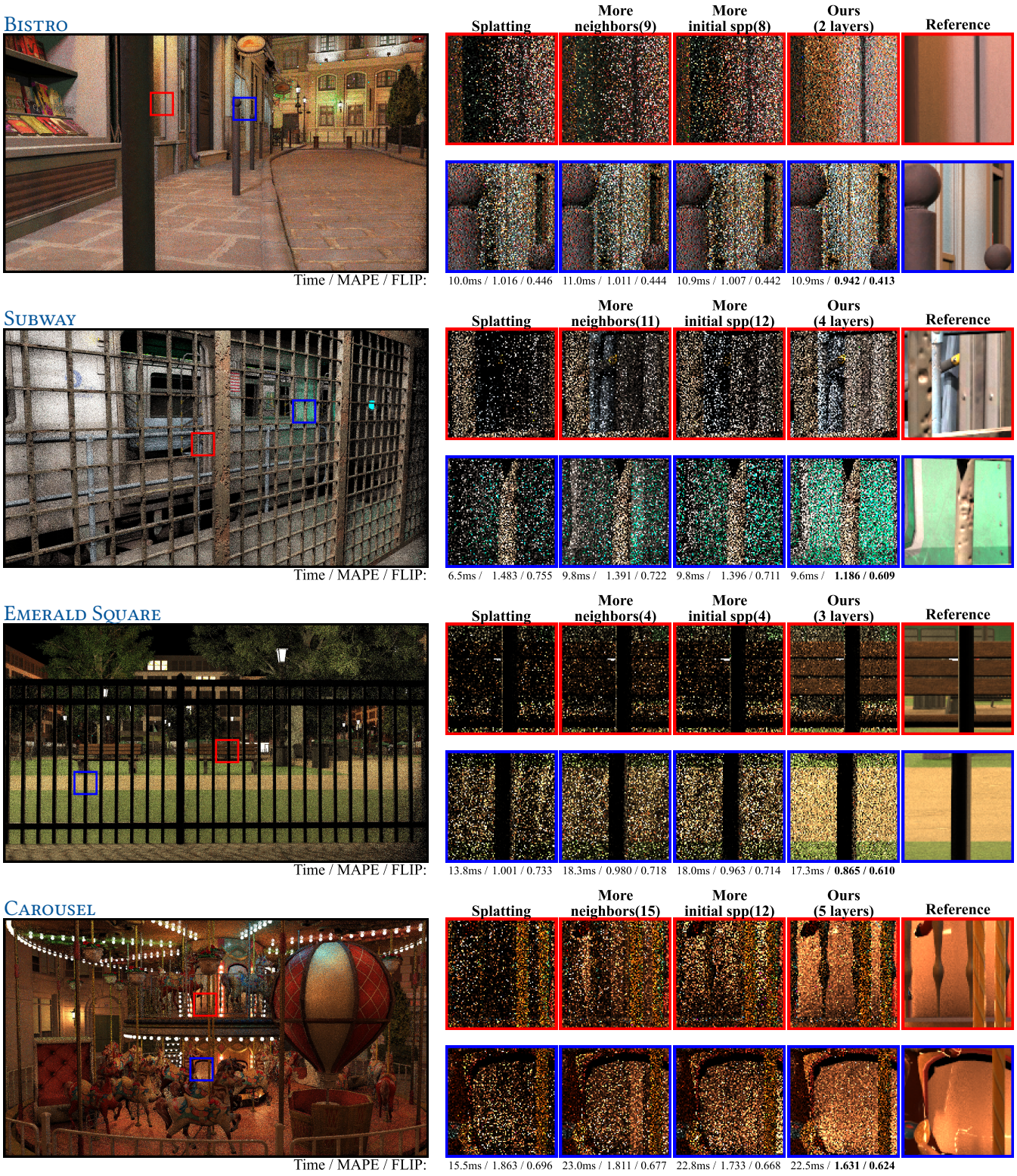

Fig. 9. Comparison between our method and the original and two variants of Reservoir Splatting [Liu et al. 2025] across multiple scenes. The "more neighbors" variant increases the number of spatial neighbors reused in disoccluded regions, while the "more initial spp" variant increases the number of samples per pixel in those regions. Otherwise, all methods use 1 initial candidate and 1 spatial neighbor.

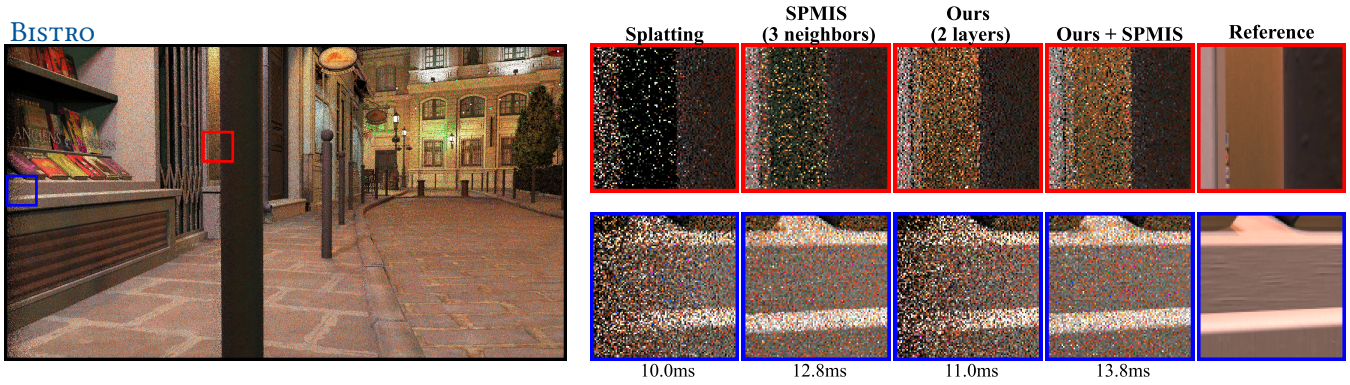

Fig. 10. Comparison between our method and SPMIS in the Bistro scene. SPMIS improves disocclusion quality when spatial reuse is effective (e.g., along screen boundaries), whereas our method better handles challenging disocclusions by leveraging temporal information. Combining the two methods yields the best overall quality.

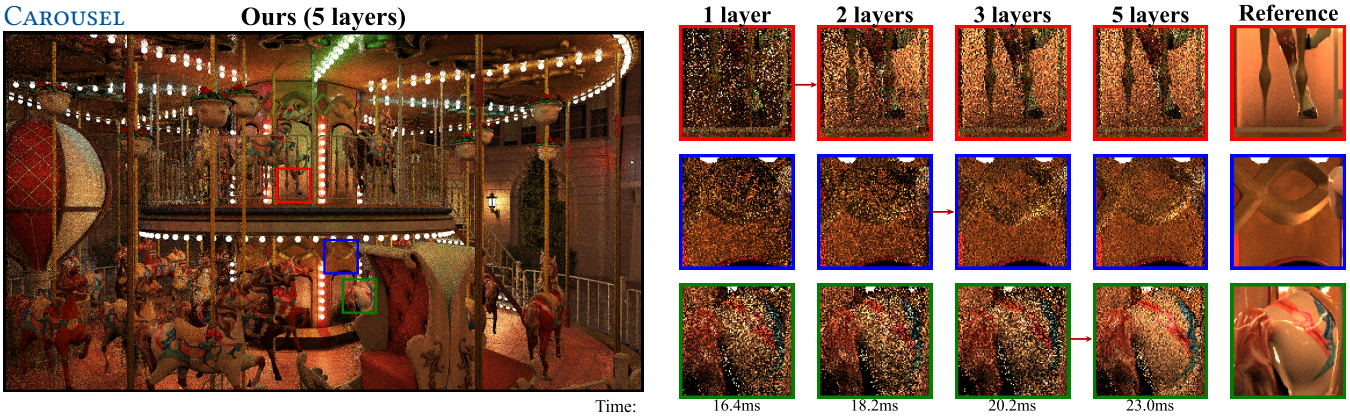

Fig. 11. Comparison of varying maximum layer counts in the Carousel scene shows that additional layers yield progressive improvements across different regions, though at a higher computational cost.