Stochastic Graph Compression for Constant-Memory Differentiable Light Tracing

Linas Beresna
Simon Fraser University
Burnaby, Canada
linas_beresna@sfu.ca

Eugene Fiume
Simon Fraser University
Burnaby, Canada
eugene_fiume@sfu.ca

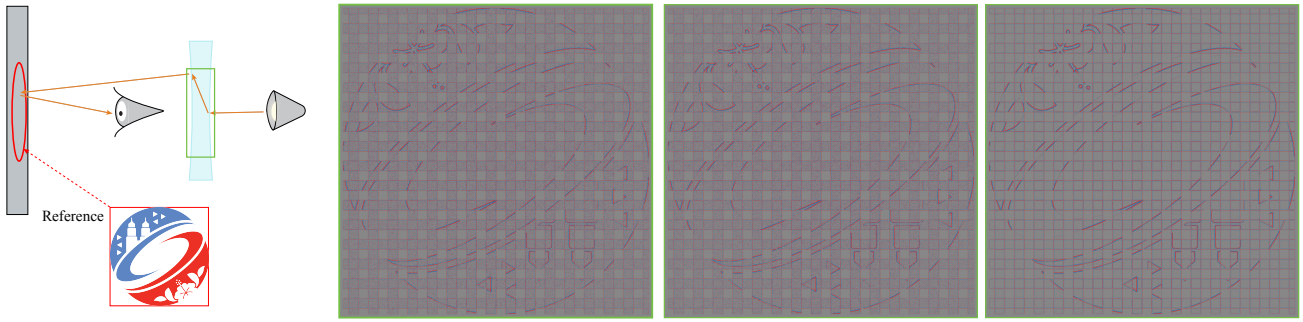


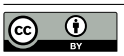
Figure 1: Constant-memory differentiable light tracing. A caustic formed by light refracting through a glass slab onto a diffuse receiver (left, with the reference target), differentiated with the attached formulation of Section 2.5. Backpropagating an image loss to the slab geometry, ResLRB and naive AD (`ptracer`) produce matching gradients at 32 spp. Beyond 32 spp naive AD exhausts GPU memory, whereas ResLRB holds memory constant and reaches 1024 spp, yielding the same gradient with less Monte Carlo noise.

Abstract

We describe *Reservoir Light Replay Backpropagation* (ResLRB), a method for reverse-mode differentiable light tracing whose memory is constant in path length and in the number of sensor connections per light path. Naive automatic differentiation of a light tracer records a computation graph that grows with the number of valid sensor connections along each path, since flux can be splatted from every scattering vertex; adjoint path replay removes the analogous dependence on depth for viewpoint path tracing, but its non-branching structure does not extend to the splatting case. We compress the adjoint graph during the primal pass by stochastically retaining a single representative sensor connection per light path via a streaming weighted reservoir. The adjoint pass reconstructs the path deterministically by replaying its pseudorandom number sequence and backpropagates only through the retained connection, yielding an unbiased gradient estimator whose peak memory is that of differentiating a single scattering event.

CCS Concepts

• **Computing methodologies** → **Ray tracing**.



SA Technical Communications '26, Kuala Lumpur, Malaysia

ACM ISBN 979-8-4007-2841-9/2026/12
https://doi.org/10.1145/3829339.3847837

Keywords

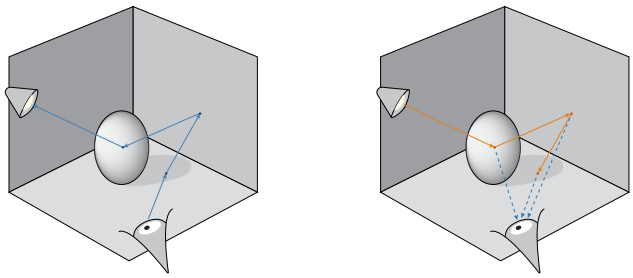
differentiable rendering, adjoint method, light tracing, constant memory, reservoir sampling

ACM Reference Format:
Linas Beresna and Eugene Fiume. 2026. Stochastic Graph Compression for Constant-Memory Differentiable Light Tracing. In *SIGGRAPH Asia 2026 Technical Communications (SA Technical Communications '26), December 01–04, 2026, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3829339.3847837

1 Introduction

Differentiable rendering computes the derivatives of a simulated image with respect to scene parameters such as materials, emitters, and geometry, enabling gradient-based optimization for inverse problems. Because the parameter domain is large, these derivatives are computed in reverse mode: reverse-mode automatic differentiation (AD) records every operation of the forward simulation into a computation graph and traverses it backwards, propagating the gradient of a scalar loss to all parameters at once. In a Monte Carlo light transport simulation the graph therefore holds every scattering event along every path and grows with path depth; for deep paths or high sample counts its footprint can exceed available GPU memory. Radiative Backpropagation [Nimier-David et al. 2020] recasts this gradient computation as a transport problem, trading memory for time.

Path Replay Backpropagation [Vicini et al. 2021] (PRB) removes that cost for viewpoint path tracing. The primal pass simulates the



(a) Viewpoint path tracing. **(b) Light tracing.**

Figure 2: Path–sensor connectivity. A viewpoint path (left) contributes through a single emitter connection, giving a non-branching graph. A light path (right) can connect to the sensor at every vertex (dashed blue arrows), so its graph grows with both depth and connection count.

path forward and stores only the seed of its pseudorandom number generator (PRNG); the adjoint pass re-seeds that PRNG and re-traces the path, so the identical sequence of random numbers regenerates the identical sequence of vertices. We call this *PRNG replay*: instead of saving the path, it calculates each point on the fly and updates gradients step-by-step. The full graph is never held in memory, giving constant memory in path depth at the cost of one additional traversal.

This method does not directly apply to light tracing. A viewpoint path typically contributes to the image once, via a single emitter connection, giving a non-branching graph; a light path instead propagates outward from the emitter and can connect to the sensor at every scattering vertex. Hence a path of depth k produces up to k separate contributions, each with its own computation graph (Figure 2). Prior adjoint light tracing has sidestepped this: Lipp et al. [Lipp et al. 2024] target view-independent object-space irradiance. Instead the image-space splatting we consider is what introduces the branching.

We describe a method that obtains reverse-mode gradients for light tracing with memory consumption that is constant in path length and in the number of sensor connections per path, preserving the two-pass structure of PRB. The primal pass retains a single sensor connection per path in a streaming weighted reservoir [Vitter 1985]; the adjoint pass reconstructs the path once by PRNG replay and backpropagates only through the retained connection, yielding an unbiased estimator. Streaming resampling has a long history in rendering for variance reduction and reuse, from Resampled Importance Sampling [Talbot et al. 2005] and ReSTIR [Bitterli et al. 2020] to path-space [Lin et al. 2022] and bidirectional variants [Hedstrom et al. 2025]; closest to our work, Nimier-David et al. [Nimier-David et al. 2022] use reservoir selection in differentiable volume rendering to sample a single distance along a ray. We apply a structurally similar mechanism along a surface light path, but to a different end: bounding the recorded computation graph rather than reducing variance.

2 Method

The name *Reservoir Light Replay Backpropagation* (ResLRB) reflects the method's three ingredients: reservoir compression of per-path sensor connections, adjoint light tracing, and path replay [Vicini et al. 2021].

2.1 Notation

A light path $\bar{x} = (x_0, \dots, x_k)$ starts from a point x_0 sampled on an emitter and propagates outward through the scene. At each scattering vertex x_i, $i \geq 1$, the path can be connected to the sensor, contributing flux C_i to a single sensor coordinate $u_i \in \mathbb{R}^2$; the per-path image-space contribution is the multiset $\{(C_i, u_i)\}_{i=1}^k$. Each C_i depends on the differentiable scene parameters θ through emitter radiance, scattering functions, the geometric term, and sensor importance. We develop the method for the *detached* case, in which u_i is independent of θ so that the only θ-dependence is through C_i, and extend it in Section 2.5 to the *attached* case, where u_i depends on θ as well.

For a scalar objective $\mathcal{L}$ defined on the rendered image, the gradient $\nabla_\theta \mathcal{L}$ is the quantity we wish to estimate. Writing $\mathcal{L}$ as a sum of per-pixel terms and pulling the per-path contribution out, the gradient associated with a single path is

$$g(\bar{x}; \theta) = \sum_{i=1}^{k} \nabla_\theta \big[\, f_i(C_i(\theta),\, u_i) \,\big], \tag{1}$$

where f_i is the pixel-filter-weighted splat of C_i to coordinate u_i, treated as linear in C_i. Naive reverse-mode AD records one computation graph per term and traverses all k in the backward pass; our goal is to estimate $g(\bar{x}; \theta)$ while recording at most one at a time.

2.2 Stochastic Selection of One Connection

During the primal pass, we stream the candidate connections $\{(C_i, u_i, w_i)\}_{i=1}^k$ into a single-item weighted reservoir, where $w_i = \ell(C_i) \geq 0$ is a scalar weight. Any ℓ that is strictly positive wherever a connection contributes leaves the estimator of Section 2.3 unbiased, so ℓ is free to be chosen for variance-reduction alone. We use luminance: it makes the selection probability proportional to each connection's share of the path's image contribution, concentrating the retained connection on the vertices that dominate the gradient, and it is already computed in the primal pass at no additional cost. The reservoir maintains a running sum $W = \sum_{j=1}^{i} w_j$ and replaces its current contents with the new candidate at step i with probability w_i / W, as illustrated in Figure 3. After the path terminates, the reservoir holds index $I \in \{1, \dots, k\}$ selected with probability

$$\Pr(I = i) = \frac{w_i}{\sum_{j=1}^{k} w_j} = \frac{w_i}{W_k}. \tag{2}$$

This is the standard property of weighted reservoir sampling [Vitter 1985]. Crucially, only the final W_k and the data associated with the selected index I are kept; all other candidates are discarded.

2.3 Unbiased Estimator

Given the selected index I, we define the random estimator

$$\hat{S}(\bar{x}; \theta) = \frac{W_k}{w_I} \cdot f_I(C_I(\theta),\, u_I). \tag{3}$$

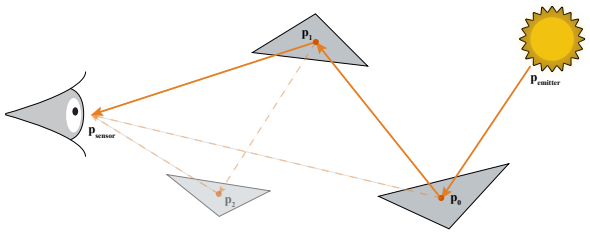


Figure 3: Stochastic selection of a representative connection. Vertices p_0, p_1, p_2 each offer a valid sensor connection (dashed). The reservoir keeps one (solid, here p_1) and discards the rest, collapsing the branching adjoint graph into a single path.

Taking the expectation over the random selection,

$$\mathbb{E}_I\left[\hat{S}(\bar{x};\theta)\right] = \sum_{i=1}^{k} \Pr(I=i)\cdot\frac{W_k}{w_i}\cdot f_i(C_i(\theta),u_i)$$
$$= \sum_{i=1}^{k}\frac{w_i}{W_k}\cdot\frac{W_k}{w_i}\cdot f_i(C_i(\theta),u_i) = \sum_{i=1}^{k} f_i(C_i(\theta),u_i), \quad (4)$$

which is exactly the deterministic per-path image contribution, so $\hat{S}$ is unbiased for the sum of all k splats. Provided f_i is sufficiently smooth that differentiation and expectation can be interchanged,

$$\mathbb{E}_I\left[\nabla_\theta \hat{S}(\bar{x};\theta)\right] = \nabla_\theta\,\mathbb{E}_I\left[\hat{S}(\bar{x};\theta)\right] = g(\bar{x};\theta), \quad (5)$$

so $\nabla_\theta\hat{S}$ is unbiased for the per-path gradient. Variance increases relative to evaluating all k terms, because every sensor connection is not computed, but the estimator remains correct in expectation.

2.4 Two-Pass Evaluation via Path Replay

The estimator $\nabla_\theta\hat{S}$ requires the AD graph for only the winning connection, and is computed by PRNG replay [Vicini et al. 2021]. In the *primal pass*, we run the light path forward, build the reservoir, and store the PRNG seed together with the selected index I and the final weight sum W_k. The splat f_I is evaluated, scaled by W_k/w_I, and accumulated into the image; no AD graph is recorded. In the *adjoint pass*, we re-seed the PRNG and re-trace the path up to depth I, enabling reverse-mode AD only for the local scattering computation at each vertex while the remaining state is reconstructed in detached form. Each vertex contributes to $\nabla_\theta\hat{S}$ through its local factor in C_I: using the cached primal value of C_I as a multiplicative weight, the contribution at vertex x_i is $\nabla_\theta f_s^{(i)}\cdot\left(C_I/f_s^{(i)}\right)$, where $f_s^{(i)}$ is the local BSDF evaluation. The log-derivative identity $\nabla_\theta \log f = \nabla_\theta f/f$ lets each vertex contribute its share of $\nabla_\theta C_I$ without holding any other vertex's graph; the scaling factor W_k/w_I depends only on primal-pass quantities and is treated as a constant.

Each adjoint iteration opens a fresh local AD scope, records the graph for one scattering event, propagates the local gradient into the scene parameters, and discards the graph before the next vertex. Peak AD memory in the adjoint pass is therefore bounded by the cost of differentiating a single scattering event, regardless of k or I. Figure 4 outlines both passes.

```
def primal(ray):
  R, W = empty_reservoir(), 0
  v = trace(ray)
  for i in range(max_depth):
    if not v.valid(): break
    C_i, w_i = connect_to_sensor(v)
    W += w_i
    if uniform() < w_i / W:
      R.store(v, C_i, w_i, i)
    v = trace(v)
  splat(R.C * (W / R.w), R.u)
  return R.depth

def adjoint(ray, I, δ_L):
  v = trace(ray)
  for i in range(max_depth):
    if not v.valid(): break
    consume_reservoir_rng()
    f_s = bsdf_eval(v)
    δ_θ += backward_grad(f_s, δ_L * R.C / f_s)
    if i == I: break
    v = trace(v)
```

Figure 4: ResLRB primal (top) and adjoint (bottom) passes.

2.5 Extension to Attached Differentiation

The formulation above is *detached*: the splatted position u_i is held independent of θ. This fails whenever perturbing θ moves the path and shifts u_i, most strongly at specular and refractive vertices, as in the glass-slab caustic of Figure 1. We extend to this *attached* case without altering the reservoir or replay structure. Parameterizing each ray segment by its endpoint coordinates $\mathbf{p}_i = (\mathbf{u}_{i-1}, \mathbf{u}_i) \in \mathbb{R}^4$, a forward-mode pass yields per-vertex Jacobians $J_i = \partial\mathbf{p}_{i+1}/\partial\mathbf{p}_i$ that chain into a path Jacobian $J_{0\to i}$, mapping how a parameter perturbation propagates along the path to move u_i, exactly the dependence the detached case discards. We cache $J_{0\to i}$ at constant extra storage when the reservoir selects a vertex.

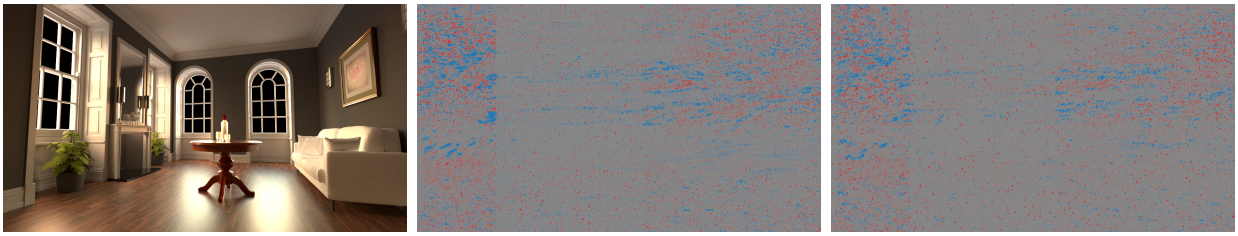
The selected connection contributes splat Jacobians $\partial u_I/\partial\mathbf{p}_I$ and $\partial C_I/\partial\mathbf{p}_I$, composed with the cached $J_{0\to I}$. During replay these are projected into each vertex's local frame through the inverse accumulated Jacobian, propagating the radiometric adjoint δ_L and positional adjoint $\delta_u = \nabla_u\mathcal{L}$ into the scene parameters; we regularize the rank-deficient Jacobians at perfectly diffuse vertices, where the inverse is otherwise ill-defined.

3 Verification

We implement ResLRB as a custom integrator in Mitsuba 3 [Jakob et al. 2022b] on top of Dr.Jit [Jakob et al. 2022a], and evaluate it against naive reverse-mode AD (Mitsuba's `ptracer`) and detached PRB on an NVIDIA RTX 5080 (16 GB) using the CUDA/OptiX backend.

Gradient correctness. Figures 1 and 5 compare gradients under naive AD and ResLRB: the two agree up to Monte Carlo noise, the single retained connection, reweighted by W_k/w_I, reproducing the full per-path contribution in expectation.

Memory and timing. Sweeping the maximum path depth (Figure 6), naive AD grows linearly and exhausts GPU memory beyond depth 16, whereas ResLRB holds memory constant at 142 MB, bounded by the cost of differentiating a single scattering event. The replay traversal is what it pays for this: at depth 4 a gradient step



Reference render Naive AD ResLRB

Figure 5: Gradient validation. Gradient of an L_2 image loss with respect to the floor's diffuse reflectance in the Grey & White Room (path length 32, 32 spp); red is positive, blue negative, grey zero. Naive AD (`ptracer`) and ResLRB agree in sign and spatial structure and differ only in Monte Carlo noise, confirming that reservoir selection leaves the estimator unbiased.

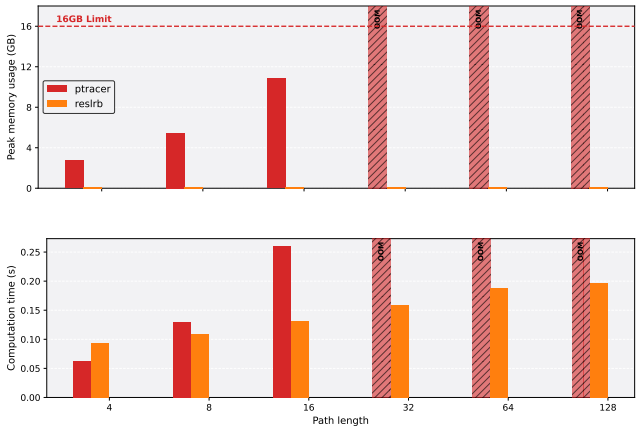


Figure 6: Memory and time versus maximum path length. Naive AD (`ptracer`) grows linearly and exhausts memory past depth 16; ResLRB's bounded graph keeps memory flat and its per-step cost nearly so.

costs 1.5× that of naive AD (93 vs. 63 ms), which needs only a single primal sweep. The overhead amortizes quickly as the memory traffic of the growing graph comes to dominate: the two are at parity by depth 8, and by depth 16 ResLRB is roughly twice as fast (132 vs. 260 ms), beyond which naive AD no longer runs. From depth 4 to depth 128 ResLRB's own cost rises only from 93 to 197 ms.

4 Conclusion

We described a two-pass adjoint formulation for differentiable light tracing whose memory is constant in path length and in the number of sensor connections per path, obtained by compressing the per-path connections into a single-element weighted reservoir, replaying the path deterministically during the adjoint pass, and opening a local AD scope per vertex. Reservoirs of capacity $m > 1$ interpolate along the memory–variance trade-off, multi-pass gathering schemes could retain every connection at the cost of further traversals, and the attached extension sketched above opens the method to geometric inverse problems. We expect each of these to build naturally on the framework described here.

Acknowledgments

We acknowledge with gratitude the funding provided for this research by Simon Fraser University as well as a Discovery Grant provided by the Natural Sciences and Engineering Research Council of Canada. Additionally, we thank Benedikt Bitterli [Bitterli 2016] for providing The Grey & White Room (from user Wig42 from blendswap).

References

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

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

Trevor Hedstrom, Markus Kettunen, Daqi Lin, Chris Wyman, and Tzu-Mao Li. 2025. ReSTIR BDPT: Bidirectional ReSTIR Path Tracing with Caustics. *ACM Trans. Graph.* 44, 5, Article 169 (Sept. 2025), 16 pages. doi:10.1145/3744898

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. *Mitsuba 3 renderer.* https://mitsuba-renderer.org.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022). doi:10.1145/3528223.3530099

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

Lukas Lipp, David Hahn, Pierre Ecormier-Nocca, Florian Rist, and Michael Wimmer. 2024. View-Independent Adjoint Light Tracing for Lighting Design Optimization. *ACM Trans. Graph.* 43, 3, Article 35 (May 2024), 16 pages. doi:10.1145/3662180

Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Trans. Graph.* 41, 4, Article 44 (July 2022), 20 pages. doi:10.1145/3528223.3530073

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). doi:10.1145/3386569.3392406

Justin F. Talbot, David Cline, and Parris Egbert. 2005. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* (Konstanz, Germany) *(EGSR '05)*. Eurographics Association, Goslar, DEU, 139–146.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4 (Aug. 2021), 108:1–108:14. doi:10.1145/3450626.3459804

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Softw.* 11, 1 (March 1985), 37–57. doi:10.1145/3147.3165