Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time

DELIO VICINI, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
SÉBASTIEN SPEIERER, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

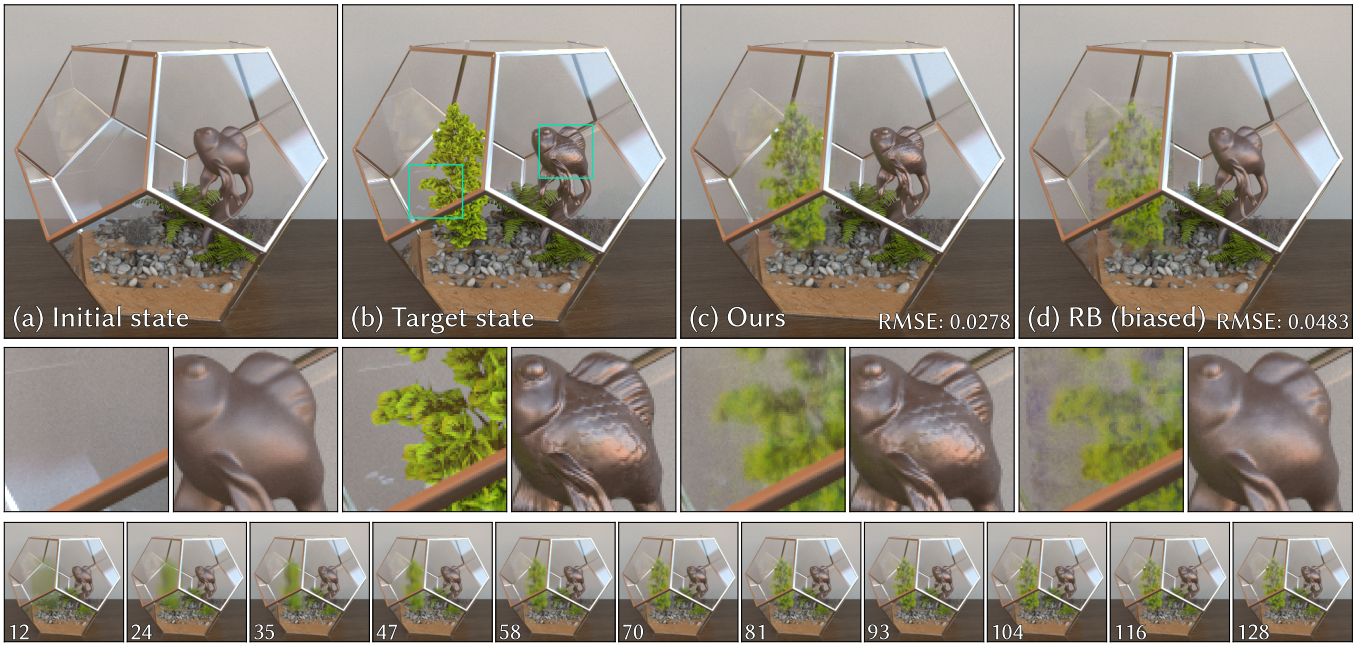


Fig. 1. Inverse reconstruction of a scene with complex lighting and heterogeneous structure. Given the initialization **(a)**, we seek to reconstruct the target **(b)** involving normal-mapped surface variation and roughness changes on the fish sculpture, and the addition of a plant based on triangular geometry. Using three rendered views of the target, we apply our proposed *path replay backpropagation* (PRB) **(c)** and a linear-time version of *radiative backpropagation* (RB) [Nimier-David et al. 2020] **(d)** to reconstruct the modified sculpture and a heterogeneous medium approximating the plant. Our method computes unbiased gradients and is able to converge to a higher-quality solution at equal time. The second and third rows show insets and PRB's convergence over time.

Differentiable physically-based rendering has become an indispensable tool for solving inverse problems involving light. Most applications in this area jointly optimize a large set of scene parameters to minimize an objective function, in which case reverse-mode differentiation is the method of choice for obtaining parameter gradients.

However, existing techniques that perform the necessary differentiation step suffer from either statistical bias or a prohibitive cost in terms of memory and computation time. For example, standard techniques for automatic differentiation based on program transformation or Wengert tapes lead to impracticably large memory usage when applied to physically-based rendering algorithms. A recently proposed adjoint method by Nimier-David et al. [2020] reduces this to a constant memory footprint, but the computation time for unbiased gradient estimates then becomes quadratic in the number of scattering events along a light path. This is problematic when the scene contains highly scattering materials like participating media.

In this paper, we propose a new unbiased backpropagation algorithm for rendering that only requires constant memory, and whose computation time is linear in the number of scattering events (i.e., just like path tracing). Our approach builds on the invertibility of the local Jacobian at scattering interactions to recover the various quantities needed for reverse-mode differentiation. Our method also extends to specular materials such as smooth dielectrics and conductors that cannot be handled by prior work.

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

Additional Key Words and Phrases: differentiable rendering, inverse rendering, radiative backpropagation, gradient-based optimization

ACM Reference Format:
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4, Article 108 (August 2021), 14 pages. https://doi.org/10.1145/3450626.3459804

Authors' addresses: Delio Vicini, delio.vicini@epfl.ch, EPFL; Sébastien Speierer, sebastien.speierer@epfl.ch, EPFL; Wenzel Jakob, wenzel.jakob@epfl.ch, EPFL.

1 INTRODUCTION

Recent progress on differentiable rendering has led to algorithms that are able to evaluate the gradient of physically based light transport simulations with respect to arbitrary scene parameters including the properties of reflectance models, procedural or bitmap textures and volumes, geometry, and camera pose. Such derivatives play an increasingly important role in the solution of inverse problems due to the vastness of the underlying parameter space. Consider that even a simple scene with a low-resolution material (e.g. 768×768 RGB texels) has over 1.7 million parameters that must all be optimized! In this high-dimensional setting, the gradient becomes crucial, since it provides a direction of steepest descent that can guide a variety of different gradient-based optimization algorithms.

Despite this encouraging progress, simulations featuring complex light transport phenomena continue to pose severe challenges: effects like global interreflection, subsurface scattering, and heterogeneous participating media produce immense amounts of unstructured arithmetic, whose effect on derivatives must be carefully tracked. The computation of derivatives must furthermore proceed in *reverse mode* [Linnainmaa 1976; Griewank and Walther 2008] to efficiently convert a small number of derivatives at the rendering algorithm's output end into a large number of derivatives at the input end. The output end typically involves a scalar objective computed from the rendered image, while the input refers to the high-dimensional scene parameter space.

The inevitable predicament faced by any task involving such a reverse-mode differentiation step is that the differential version of an algorithm requires access to certain quantities of the original (henceforth "*primal*") computation, and these accesses occur in an order that is reversed when compared to the primal calculation. Because of this fundamental ordering conflict, they normally cannot be recomputed on the fly and must be obtained in some other way. In a tiny self-contained calculation, it may be possible to simply remember all steps, but this approach does not scale: recent graphics cards can sustain up to $\sim 10^{12}$ single precision FLOPS, which works out to roughly 4 terabytes of intermediate variable state per second.

The standard remedy to this problem entails discarding most primal variables except for a sparse set of *checkpoints* [Griewank and Walther 2000] placed at strategic locations like function calls or the beginning of each n-th loop iteration. A checkpoint captures the full program state at an instant of time, enabling recovery of discarded information via re-computation in a classic computation versus memory trade-off. However, even individual checkpoints tend to be large in the context of rendering, and accessing them incurs significant storage costs and memory access latencies.

An intriguing way to side-step the overheads of program reversal via memorization is to actually perform the computation in reverse: all problems discussed above will simply disappear if the individual operations comprising a calculation can be evolved backwards in time [Griewank and Walther 2000, Ch. 4.3]. The adjoint sensitivity method [Pontryagin 1962] from the area of optimal control is the classical example of this idea: its primal phase integrates an ordinary differential equation up to a certain point in time. Differentiation then follows the same trajectory in reverse by taking negative timesteps starting from the endpoint.

Nimier-David et al. [2020] recently introduced a new approach to differentiable rendering termed *Radiative Backpropagation* (RB) that follows this high-level motivation. The authors observe that differentiable rendering is equivalent to the solution of a reversed light transport problem, where gradients are emitted by the camera, scattered by scene objects, and eventually received by objects with differentiable scene parameters. This problem can be solved directly without memorizing intermediate state, which removes the memory footprint issue and also improves efficiency.

However, this work has two main limitations: RB is either biased or requires a recursive radiance estimate at each scattering interaction in addition to its own recursion, making its computation time *quadratic* in the number of scattering events along a light path. The method also cannot differentiate interactions involving perfectly specular BSDFs, such as smooth conductors or dielectrics.

We propose *Path Replay Backpropagation* (PRB) to address both limitations. Our method splits gradient evaluation into two separate passes: in a first step, we sample light paths as usual and record a small amount of information about each complete light path: in the simplest case, this is just the path's contribution to the (hypothetical) image. Then, we reset the pseudorandom number generator and recompute the *same light paths* once more, while backpropagating derivatives to scene parameters. The key idea of our approach is that certain steps of the computation can be inverted. By taking advantage this invertibility and the data stored by the preceding pass, we are able to recover all information required by the adjoint phase on the fly.

The time complexity of the resulting algorithm has a linear dependence on path length, while computing the same unbiased derivatives as conventional AD with dramatically reduced memory usage. The two-pass scheme can furthermore generalize to derivatives of specular materials, requiring only a small amount of extra storage (a 4×4 matrix per sample) that remains independent of path length. PRB is the first method that can efficiently compute unbiased reverse-mode derivatives of volume transport simulated using delta tracking. In summary, our contributions are:

- a linear time, unbiased method for differentiable path tracing.
- a generalization of that algorithm to specular materials.
- the ability to efficiently differentiate unbiased volume transport based on delta tracking.

The proposed PRB method and its predecessor (RB) leverage complementary notions of reversibility: RB exploits the Helmholtz reciprocity of differential light transport to compute arbitrarily many derivatives using a single differential transport phase, while PRB exploits mathematical invertibility to avoid redundant computation and storage. It is the combination of both types of reversibility that finally opens the door to efficient and unbiased reverse-mode derivatives of physically-based light transport.

2 RELATED WORK

Differential light transport in other fields. Derivatives of Monte Carlo simulations that are highly related to recent work on differentiable rendering have been used to model the criticality of nuclear reactors [Lux and Koblinger 1990], and to perform inverse modeling

of tissue [Hayakawa et al. 2001]. These types of methods are named *Perturbation Monte Carlo* or *Differential Monte Carlo*.

Delta tracking [Woodcock et al. 1965] and similar approaches based on null-collisions are widely used to sample the radiative transfer equation, and discrete decisions in this method pose additional challenges during differentiation. A solution to this problem was studied in the area of physics [Tregan et al. 2019] but remains limited to forward-mode differentiation.

Forward path derivatives. Various prior works in computer graphics have differentiated light paths in a forward sense, propagating an infinitesimal perturbation of a scene or ray parameter through the calculation to determine how it influences the rendered image or outgoing ray. Such derivatives find use in texture filtering of objects observed through specular reflection [Igehy 1999], when interpolating computed diffuse [Ward and Heckbert 1992] or specular illumination [Chen and Arvo 2000], and for adaptive sampling and reconstruction [Ramamoorthi et al. 2007]. They can also be used to determine valid specular path configurations to render effects such as caustics or glints [Mitchell and Hanrahan 1992; Jakob and Marschner 2012; Zeltner et al. 2020]. Derivatives have also been used in image space, for example to render gradient images that are subsequently reconstructed [Hua et al. 2019].

Our generalization of RB to the specular case involves forward derivatives of materials to enable gradient-based optimization of the index of refraction and shape of smooth dielectric or conducting objects. The mathematics of this process are related to prior work on specular path derivatives and path manifolds [Igehy 1999; Chen and Arvo 2000; Jakob and Marschner 2012].

Differentiable rendering. Inverse problems in graphics and vision require methods that are able to optimize an objective using scene parameter derivatives, which often involves differentiable rasterization of meshes or volumes [Loper and Black 2014; Rhodin et al. 2015; Kato et al. 2018; Liu et al. 2019; Petersen et al. 2019; Laine et al. 2020]. These methods are restricted to primary visibility and ignore indirect effects like shadows, interreflection and depth of field.

Differentiation of the full rendering equation [Li et al. 2018; Nimier-David et al. 2019] is required when the optimization task at hand involves such indirect effects. This is also the case for participating media that are often characterized by significant multiple scattering [Gkioulekas et al. 2013; Khungurn et al. 2015; Gkioulekas et al. 2016; Zhao et al. 2016; Zhang et al. 2019; Che et al. 2020].

Khungurn et al. [2015] compute forward-mode derivatives to optimize a 12-dimensional material appearance model, reusing the contribution of a complete light path in a way that resembles our two-pass scheme for non-specular interactions. In contrast, our focus is on techniques that are compatible with reverse-mode derivative propagation to simultaneously optimize millions of parameters.

Geometric discontinuities. Naïve differentiation of illumination integrals yields incorrect gradients when differentiated parameters influence the location of discontinuities (e.g. object silhouettes). Li et al. [2018] proposed the first method to correctly account for this effect via edge sampling. Later works have proposed more efficient ways of addressing this issue by converting boundary integrals into interior integrals [Loubet et al. 2019; Bangaru et al. 2020] or sampling on a higher dimensional path space [Zhang et al. 2020].

Our paper studies ways of efficiently differentiating light paths with many interactions, and we ignore discontinuities that would only distract from this task. That said, reverse-mode propagation of gradients can be combined with the change of variables approach of Loubet et al. [2019], and a concurrent publication [Zeltner et al. 2021] explores aspects of such a combination.

Reversible programming. Time-reversibility has been used to reduce the costs of differentiation in fluid dynamics [McNamara et al. 2004], robotics [Hoshyari et al. 2019], and reversible residual networks in machine learning [Gomez et al. 2017; Chen et al. 2018]. Invertible neural network architectures can further be employed to learn parametric sampling methods [Dinh et al. 2015, 2016].

Reversible computation also appears in quantum theory, which models the temporal evolution of a system using unitary operators that are inherently reversible. Shimada and Hachisuka [2020] recently proposed a Monte Carlo sampling algorithm that is constructed in this way to be realizable on quantum computers.

Our approach is similar to reversible programming methods, but instead of explicitly running an inverse program, it relies on the invertibility of the (low-dimensional) Jacobian relating the positions of adjacent scattering interactions.

Automatic differentiation. Our method relies on *automatic differentiation* (AD) for derivative evaluation. AD comes in two main flavors: *forward mode* propagates derivatives in program order and is ideal when the function to be differentiated has many outputs and few inputs. *Reverse mode* is the diametric opposite and can be challenging to use as outlined earlier. Our algorithm uses both directions internally, which makes it an example of a *mixed mode* method. Beyond the choice of directionality, AD covers a surprisingly wide spectrum of methods owing to the diverse needs of scientific software. Approaches range from simple dual number representations [Siskind and Pearlmutter 2008] to compiler-like frameworks that perform dataflow analyses on intermediate program representations [Hascoet and Pascual 2013]. The underlying principles were pioneered decades ago and are now the focus of renewed attention thanks to the growing use of gradient-based optimization. We refer to the book of Griewank and Walther [2008] that provides an excellent survey of this area.

3 BACKGROUND

We begin by reviewing relevant background material covering differentiable rendering and radiative backpropagation.

3.1 Differentiable Rendering

Physically-based rendering techniques estimate high-dimensional integrals on the space of light paths [Kajiya 1986] to compute intensities I_j of pixels $j = 1, \dots, n$ defined as

$$I_j(\boldsymbol{\pi}) = \int_{\mathcal{P}} f_j(\mathbf{x}, \boldsymbol{\pi}) \, \mathrm{d}\mathbf{x}. \tag{1}$$

Here, the *scene parameters* $\boldsymbol{\pi}$ encode a representation of the virtual scene including the configuration of material and geometry, $\mathcal{P}$ denotes the space of all light paths, $\mathbf{x}$ is an individual path and f_j

quantifies the contribution of this path towards a pixel measurement j. In contrast, *inverse rendering* techniques solve a general minimization problem of the form

$$\hat{\pi} = \arg\min_{\boldsymbol{\pi}} g(\mathbf{I}(\boldsymbol{\pi})), \tag{2}$$

where g is a differentiable objective function that takes the full image as input. In the simplest case, g might quantify the L_2 difference between $\mathbf{I}(\boldsymbol{\pi})$ and a reference image. In more complex cases, it could be a black-box nonlinear function such as a neural network. For notational convenience, the remainder of this paper will only focus on such optimizations with respect to a single scene parameter π, though all techniques developed will also support simultaneous optimization involving vast numbers of parameters.

Differentiable rendering techniques approach this challenging minimization problem using gradient-based optimization and therefore require the ability to differentiate the full process of image formation:

$$\partial_\pi I_j(\pi) = \partial_\pi \int_{\mathcal{P}} f_j(\mathbf{x}, \pi)\, \mathrm{d}\mathbf{x} = \int_{\mathcal{P}} \partial_\pi f_j(\mathbf{x}, \pi)\, \mathrm{d}\mathbf{x}, \tag{3}$$

where $\partial_\pi := \partial/\partial\pi$. The second equality above moves the partial derivative into the integrand, which is allowed if the integrand is continuous. It also holds when the integrand contains discontinuities (e.g. visibility-induced silhouette edges), and when their placement within the integration domain is furthermore independent of the scene parameter π being differentiated [Loubet et al. 2019]. This case arises for example when the optimization targets participating media or the material properties of surfaces, while leaving the geometric representation of the scene fixed. In this article, we focus on efficient reverse-mode differentiation and disregard discontinuity handling via reparameterization or edge sampling—in other words, we assume that the above equality holds.

Equation (3) further assumes that the derivatives are taken with respect to the integrand alone. However, integrals in computer graphics are almost exclusively evaluated using Monte Carlo importance sampling techniques that are typically constructed using techniques like inverse transform sampling [Devroye 1986] that effectively reparameterize the integral. This poses the question of whether this reparameterization should be applied following differentiation, or whether it should be applied at first and differentiated along with the integrand. Because the sampling strategy may potentially produce different distributions depending on the parameter π, samples from the latter approach can be understood to smoothly follow the motion of the underlying sampling strategy with respect to perturbations in π, which influences the variance properties of the resulting estimators. Since they produce samples that are attached to the underlying distribution, we refer to the latter as *attached sampling strategies* and conversely denote the former case as *detached sampling strategies*.

A concurrent article [Zeltner et al. 2021] examines the theoretical and practical ramifications of these choices, and it also discusses handling of reparameterizations in a reverse-mode context. In contrast, the present article explains how both detached and attached sampling strategies can be evaluated using only a constant memory footprint and linear time complexity.

3.2 Radiative Backpropagation

Naive application of automatic differentiation (AD) to a light transport simulation leads to extremely high memory usage due to the need to store intermediate steps of the primal computation that are later accessed during reverse-mode differentiation. Nimier-David et al. [2020] instead exploit properties unique to light transport and turn reverse-mode derivative propagation into an independent simulation that transports derivative radiation from sensors to objects with differentiable parameters. We briefly review how this works and point out issues of this transformation addressed by our work.

We focus on the surface-only case for simplicity, where the light transport in a scene satisfies the following energy balance equation [Kajiya 1986]:

$$L_o(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} L_i(\mathbf{x}, \boldsymbol{\omega}') f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\boldsymbol{\omega}'^{\perp}. \tag{4}$$

Here, L_o, L_e, and L_i refer to the outgoing, emitted, and incident radiance, and f_s is the *bidirectional scattering distribution function* (BSDF). The product of incident radiance and BSDF is integrated over projected solid angles $\boldsymbol{\omega}'^{\perp}$.

Nimier-David et al. then apply the derivative ∂_π to both sides of this equation, which produces several terms via the product rule:

$$\begin{aligned}\partial_\pi L_o(\mathbf{x}, \boldsymbol{\omega}) = \partial_\pi L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{S^2} &\partial_\pi L_i(\mathbf{x}, \boldsymbol{\omega}')\ f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}') \\ &+ L_i(\mathbf{x}, \boldsymbol{\omega}')\ \partial_\pi f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\boldsymbol{\omega}'^{\perp}.\end{aligned} \tag{5}$$

This equation can be interpreted as another kind of energy balance that now involves *differential radiance* $\partial_\pi L_i$, $\partial_\pi L_e$, and $\partial_\pi L_o$ (a separate set of such functions exist for every scene parameter!) Inspecting the various terms, this differential radiance can then be seen to satisfy the following properties:

(1) Differential radiance is emitted when the primal emission L_e depends on π.
(2) Ordinary radiance generates differential radiance when it interacts with objects, whose BSDF depends on π.
(3) Finally, once created, differential radiance scatters like ordinary light (i.e. involving the BSDF of scene objects).

Instead of performing a large number of differential transport simulation to compute a gradient image $\partial_\pi \mathbf{I}$ per scene parameter π, and then propagating these through the objective function g, Nimier-David et al. follow an adjoint approach: starting with a single gradient image $\boldsymbol{\delta}\mathbf{I}$ encoding the derivative of g, they solve equation (5) in reverse by exploiting the reciprocity of the scattering and transport operators [Veach and Guibas 1995]. In this case, only a single differential simulation is necessary. However, two issues become apparent when further scrutinizing their approach:

(1) The differential scattering equation (5) references the *primal incident radiance* L_i, coupling the primal and differential light transport problems so that both must now be solved at the same time. The standard approach for evaluating integral equations using recursive Monte Carlo sampling then requires recursion to handle both ∂L_i *and* L_i. This double recursion leads to a quadratic growth in the amount of computation as the maximum paths length increases, which can be very costly when the scene involves significant multiple scattering.

While recursion could be restricted by probabilistically choosing only one of the two terms, there is no obvious sampling scheme to do so. Finally, variance in such a stochastic scheme would increase exponentially due to this repeated sub-optimal choice.

(2) The method of Nimier-David et al. does not support materials containing Dirac delta functions (e.g. ideal specular BSDFs like smooth glass or metal surfaces). This case involves additional challenges that arise from coupling within *specular chains*, which refer to uninterrupted sequences of vertices along a light path involving such ideal specular materials.

Biased RB. Nimier-David et al. suggest that to avoid quadratic time complexity, one can simply set the incident radiance L_i to 1 during differentiation. They argue that the resulting bias is innocuous because gradients will still have the correct component-wise *sign* and can thus be expected to converge to a local minimum when combined with robust gradient-based optimization techniques. However, we find that both of these claims are generally incorrect: the sign may not match, and convergence is then also not guaranteed. In particular, we found that the sign is only correct in a very local sense for exactly the term $L_i(\mathbf{x}, \boldsymbol{\omega}')\, \partial_\pi f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')$, in which the incident radiance specifies a positive multiplicative factor. This means that the sign of this product is preserved if the radiance is set to a value of one. In practice, many such local gradients are accumulated due to the global nature of light transport and scattering, and the combination of these various signed quantities is not guaranteed to result in a sum that still has the correct sign.

Figure 2 demonstrates this issue by optimizing an enclosed object. In this case, emitter sampling is not available and the method must rely on the incident radiance term that was assumed to equal 1. This experiment shows that bias due to this approximation breaks the convergence even in a simple unimodal 1D optimization problem. Our method produces unbiased estimates and converges as expected, while retaining the linear time complexity of biased RB.

Note that we use the color scheme of Figure 2 throughout the paper: gray, blue, and red denote zero, positive, and negative-valued regions, respectively.

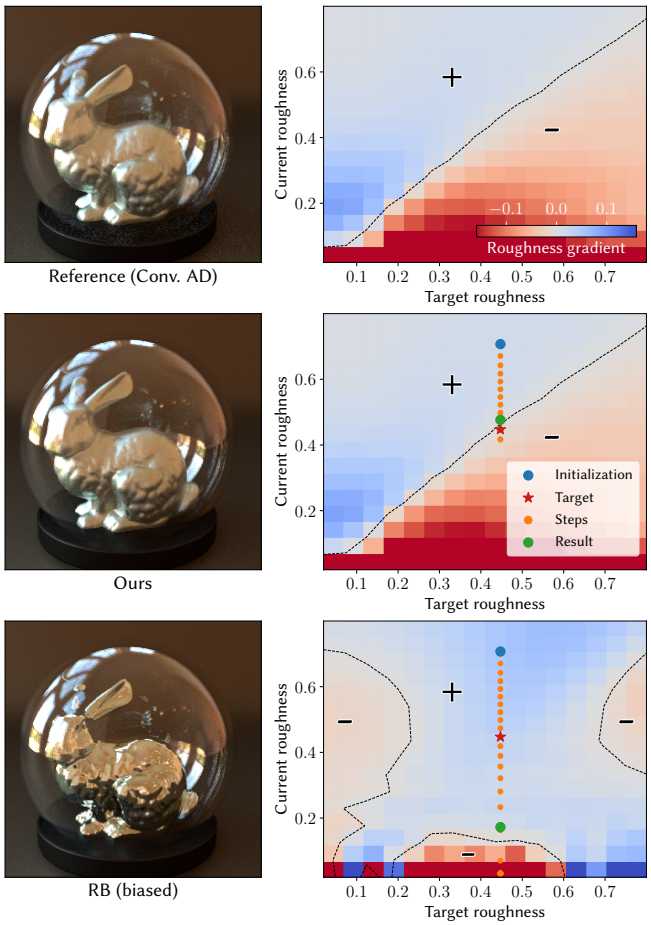


Fig. 2. Analysis of Bias in Radiative Backpropagation (RB) [Nimier-David et al. 2020]. We optimize the scalar roughness of an indirectly illuminated metal bunny contained inside a glass container. The left column shows the reference configuration and optimization results obtained by our method and biased RB. Not only did RB not converge here: it *cannot* converge regardless of initialization. Due to the scalar objective and 1D parameter space, we can exhaustively plot gradients for a range of initial and target roughness values (right column). A zero-valued gradient is expected on the diagonal, where initial and target roughness coincide. We also plot intermediate optimization steps for the optimization task on the left. These visualizations show that the gradients computed by our method match the result of conventional AD. Biased RB is not only biased but also produces incorrect signs throughout the parameter space. Optimization will therefore fail even when initialized with the correct solution.

4 METHOD

We now explain the principles of our method, starting with the basic idea that we then expand into a more general principle to address the limitations noted in Section 3. We first focus on the *detached* case and postpone the more complex *attached* case to Section 4.2.

4.1 Replaying light paths to estimate derivatives

While Monte Carlo estimators are in principle random, real-world usage relies on deterministic pseudorandom number generators that can be rewound or reinitialized to produce an identical stream of variates. This turns out to be surprisingly useful because it enables running two variants of an algorithm that will perform an identical random walk. At the same time, these two algorithms can then use the generated path vertices for different purposes.

We build on this idea to run a regular forward path tracing pass that is followed by an adjoint pass visiting the same sequence of vertices. The first phase will determine the total amount of radiance accumulated by the path, and the second "random" walk can then exploit this information to precisely reconstruct the incident illumination at each path vertex. Our method produces the same result as automatic differentiation, while using a constant amount of storage and retaining the linear time complexity of regular forward path tracing. The results do not just match conventional AD in expectation: they produce the same noise pattern, with only small numerical differences related to the different evaluation in terms of IEEE-754 floating point arithmetic.

Our method is best explained using a few lines of pseudocode. To reduce the idea to its core, we will assume that emitters are static, and we will also work without standard optimizations like MIS or next event estimation. The following code fragment specifies a standard unidirectional path tracer to clarify all notation. It takes a ray as input and returns the resulting accumulated radiance. The code is heavily simplified and focuses on what is relevant for differentiation: changes to the path throughput β and the total radiance L accumulated along the path.

```
1 def sample_path(ray):
2     L = 0, β = 1
3     for i in range(N):
4         L += β * L_e(...)
5         ω', bsdf_value, bsdf_pdf = sample_bsdf(...)
6         β *= bsdf_value / bsdf_pdf
7     return L
```

Listing 1. Path tracer sketch, where β and L denote throughput and accumulated radiance. L is also an input of the adjoint pass discussed next.

The subsequent adjoint phase has the purpose of back-propagating *adjoint radiance* δL along the same light path: this is the derivative of radiance with respect to the optimization's objective function, which captures how the radiance along this specific path should change to improve the objective's current value. To do so, we must sample the terms of the differential transport problem outlined in Equation 5. Due to the simplifying assumption of static emitters, this equation further reduces to

$$\partial_\pi L_o(\mathbf{x}, \boldsymbol{\omega}) = \int_{S^2} \partial_\pi L_i(\mathbf{x}, \boldsymbol{\omega'})\ f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega'}) + L_i(\mathbf{x}, \boldsymbol{\omega'})\ \partial_\pi f_s(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega'})\ \mathrm{d}\boldsymbol{\omega'}^{\perp}. \tag{6}$$

To recapitulate, the first term of this equation states that differential radiation propagates according to a standard random walk, and the adjoint phase simply handles this part using a loop. The second term must be handled specially and requires backpropagating the product of adjoint and incident radiance into the BSDF f_s.

The function `backward_grad(<expr>, grad_out)` evaluates the reverse-mode derivative of the expression `<expr>` to propagate a gradient with respect to the expression's output (`grad_out`) towards the scene parameters, returning another gradient resulting from this step. The following code fragment then realizes all steps of this adjoint phase:

```
1 def sample_path_adjoint(ray, L, δL):
2     β = 1
3     for i in range(N):
4         L -= β * L_e(...)
5         ω', bsdf_value, bsdf_pdf = sample_bsdf(...)
6         δ_π += backward_grad(bsdf_value, δL * L / bsdf_value)
7         β *= bsdf_value / bsdf_pdf
8     return δ_π
```

Listing 2. The adjoint phase takes L as input and reverses certain operations in `sample_path()` to propagate adjoint radiance δL to scene parameters.

Line 3 of the adjoint pass reconstructs the incident illumination at the current vertex by *subtracting* emission, if present. This is the inverse of line 4 in Listing 1. The resulting modified value L still includes the `bsdf_value` / `bsdf_pdf` ratio that was applied in line 6 of Listing 1.

Our goal is now to importance sample the term $L_i(...)\ \partial f_s(...)$ of the differential scattering equation (6), for which we reuse the existing sample $\boldsymbol{\omega'}$ with density `bsdf_pdf`. This entails dividing out the BSDF value but retaining the reciprocal density and backpropagating the product of this quantity with the adjoint radiance δL into the scene parameters corresponding to the current BSDF f_s.

Several aspects of this algorithm are remarkable: it is mathematically equivalent to reverse-mode AD that normally requires a complete program reversal including ample storage of intermediate values. Yet, the loop of our adjoint pass retains its order, exploiting the repeatability of the random walk along with a minimal use of extra information (`L`) computed in a prior phase.

Both next event estimation and multiple importance sampling are easily integrated into this framework by carefully reconstructing the direct illumination at each vertex and performing the inverse of the primal operations that are needed to accomplish this.

We note that Listing 2 involves a subtraction, which can lead to floating point cancellation errors. This case could arise when a path vertex has an extremely high contribution that dominates that of other path vertices. When processing this vertex in the adjoint pass, cancellation could cause L to round to zero. Arguably, this corresponds to a situation where the numerical accuracy of the primary simulation is also suspect.

4.2 Attached Sampling Strategies

While the previously discussed method works well in a wide range of situations, it cannot compute parameter derivatives related to perfectly specular surfaces, as this would entail backpropagating gradients through Dirac delta functions. This prevents optimizing index of refraction, geometry, or surface normals of smooth conductors and dielectrics, including interesting applications like caustic design or the inverse reconstruction of transparent objects [Lyu et al. 2020].

Instead of differentiating the BSDF alone, we must consider its behavior in relationship to the underlying sampling strategy and track how perturbations of relevant scene parameters will influence this directional sampling process. Sampling strategies in computer graphics are normally based on inverse transform sampling that can be understood as a change of variables to new coordinates $\mathbf{u} \in \mathcal{U}$ parameterizing the integration domain $\mathcal{P}$ via a mapping $T : \mathcal{U} \to \mathcal{P}$, where $\mathcal{U} = [0, 1]^n$ is a unit-sized hypercube of suitable dimension. The mapping $\mathbf{x} = T(\mathbf{u})$ is constructed from a target density $p(\mathbf{x})$ so that its Jacobian determinant satisfies $|J(\mathbf{u})| = p(\mathbf{x})^{-1}$. The re-parameterized integral then takes the form

$$I_j(\pi) = \int_{\mathcal{U}} f_j(T(\mathbf{u}), \pi)\, |J(\mathbf{u})|\, \mathrm{d}\mathbf{u} = \int_{\mathcal{U}} \frac{f_j(T(\mathbf{u}), \pi)}{p(T(\mathbf{u}))}\, \mathrm{d}\mathbf{u}. \tag{7}$$

Monte Carlo estimates of this integral will achieve low-variance if $p(\mathbf{x}) \approx f(\mathbf{x})$. Good sampling strategies must necessarily depend on the scene parameters π to accomplish this by specializing the target density and mapping to the properties of a given scene. We will add an extra dependence on π to p and T to capture this aspect.

Differentiating the re-parameterized integral yields an *attached* differential sampling strategy, where samples are geometrically

attached to the motion of $T(\mathbf{u}, \pi)$ with respect to perturbations of π:

$$\partial_\pi I_j(\pi) = \int_{\mathcal{U}} \partial_\pi \left[\frac{f_j(T(\mathbf{u}, \pi), \pi)}{p(T(\mathbf{u}, \pi), \pi)} \right] \mathrm{d}\mathbf{u}. \tag{8}$$

Applying the ideas of Section 4.1 poses additional challenges compared to the previously discussed detached case. Consider a perturbation of the shading normal or a material parameter at a given surface interaction: this change will further propagate, influencing the geometry of the remainder of this path. This will in turn also change the value of subsequent BSDF and emission terms. It is important to correctly account for these non-local dependencies during differentiation to ensure unbiased gradient estimates. Fortunately, it remains possible to follow the same core idea to develop similar methods for the attached case.

To motivate our approach, let us apply the partial derivative to the various terms in Equation (8):

$$\partial_\pi I_j(\pi) = \int_{\mathcal{U}} \frac{\partial_\pi f_j(T(\mathbf{u}, \pi), \pi)}{p(\ldots)} - \cdots \mathrm{d}\mathbf{u}, \tag{9}$$

where we have applied the quotient rule and only focus of the numerator of the first resulting term. Using the vectorial chain rule, this can be seen to equal

$$= \int_{\mathcal{U}} \frac{\partial_\pi f_j(\ldots) \; + \; \partial_{\mathbf{x}} f_j(\ldots) \cdot \partial_\pi T(\ldots)}{p(\ldots)} - \cdots \mathrm{d}\mathbf{u}. \tag{10}$$

This derivation shows how the numerator splits into a term that tracks the change of the primal integrand with respect to π, and the following product of two Jacobian matrices expresses how changes in the parameterization influence f_j due to its dependence on the light path vertices $\mathbf{x}$. Section 1 of the supplemental material contains further detail on these derivatives and their relationship to differential radiance. These transform-related derivatives constitute the main change due to attached sampling strategies, and our approach will be to similarly reconstruct them on the fly using a constant amount of precomputed information generated by a prior phase.

In contrast to the previous algorithm, we therefore not only need to track quantities related to the total accumulated radiance and throughput, but also to the path geometry. In each iteration, we spawn a ray according to the BSDF sampling strategy, and the computed derivatives must then take into account how this new ray will depend on the previous ray.

Suppose that $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ are subsequent vertices on a light path, where $\mathbf{p}_2$ was sampled given an incident ray $\mathbf{p}_0 \to \mathbf{p}_1$, as illustrated in Figure 3. We must then compute the Jacobian capturing the differential relationship $\partial \mathbf{p}_2 / \partial \mathbf{p}_0$ of these path vertices. Performed naïvely, this would be a 6×6 matrix. However, as we are only interested in rays originating on surfaces (or the camera), and directions are normalized, we can reduce it to a 4×4 matrix by switching to a 2D parametrization using coordinates that leverage the available tangential basis vectors at path vertices.

It is interesting to note that these derivatives can in principle be computed using any kind of parameterization of the space of light paths and rays, and we initially used a position-angle parameterization of rays. In this case, the Jacobian matrices involve components with incompatible units and different resulting scales, which can cause poor numerical conditioning. Switching to a position-position parameterization addressed this issue.

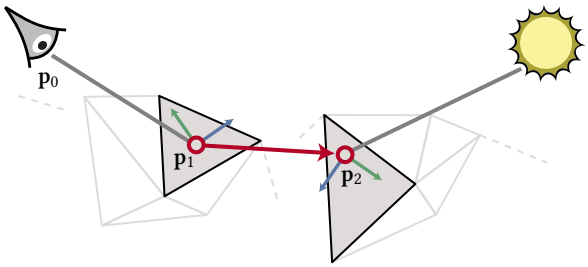


Fig. 3. The *attached* version of our method must consider how the geometry of a light path changes with respect to infinitesimal perturbations of scene parameters. Our method does so by tracking the differential relationship of a subsequent path vertex $\mathbf{p}_2$ generated from an incident path segment $\mathbf{p}_0 \to \mathbf{p}_1$ using the combination of BSDF sampling routines and ray tracing.

Pseudocode. Listing 3 contains the pseudocode of our attached gradient evaluation technique following the same conventions previously outlined in Listings 1 and 2. Each loop iteration uses forward-mode differentiation (via a function `forward_grad()`) to capture the differential relationship of adjacent path vertices and reverse-mode differentiation (via a function `backward_grad()`) to backpropagate adjoint radiance into the scene parameter gradient.

Concretely, the function `backward_grad(x, δ)` propagates the gradient of the function's output to the input parameters by evaluating the Jacobian product $\mathbf{J}_{\mathbf{x}}^{\mathsf{T}} \delta$ using automatic differentiation. On the other hand, the function `forward_grad()` takes an input variable and several output variables and computes the Jacobian matrices of all the output variables with respect to the input. This function returns Jacobian matrices, while `backward_grad()` internally multiplies the Jacobian with vector. By $\mathbf{J}_\circ$ we denote Jacobian matrices of some output variable $\circ$ with respect to the current "ray". Here, a ray is represented by the tuple of UV coordinates at its origin and intersection location. For RGB values, e.g., the accumulated radiance L, these Jacobians have 3 rows and 4 columns. The implementation keeps track of the derivative of the accumulated radiance, path throughput and the next "ray". We denote these quantities by $\mathbf{J}_L, \mathbf{J}_\beta$ and $\mathbf{J}_{\text{ray}}$. For example, $\mathbf{J}_{\text{ray}}$ contains the derivatives of the UV coordinates at bounce i and $i+1$, $(u_i, v_i, u_{i+1}, v_{i+1})$, with respect to the camera ray, represented by the tuple (u_0, v_0, u_1, v_1). The first pass then returns the accumulated radiance L and its directional derivative $\mathbf{J}_L$ to be used in the adjoint phase.

Note that automatic differentiation is only used *within* the loop body, and does not need to build an AD graph over loop iterations. Our explicitly computed Jacobian matrices account for all derivative information that is needed across loop iterations.

Near-specular scattering. We mainly motivated the attached case from the viewpoint of handling ideally specular interactions, but this approach may also be beneficial for *near-specular* scattering involving materials with only a small amount of roughness. The variance reduction properties of attached estimators are analyzed in a concurrent article [Zeltner et al. 2021].

```
def sample_path(ray):
    L = 0, β = 1
    J_L = 0_{3,4}, J_β = 0_{3,4}, J_ray = I_4
    for i in range(N):
        L += β * L_e(...)
        wo, bsdf_value, bsdf_pdf = sample_bsdf(...)
        bsdf_weight = bsdf_value / bsdf_pdf
        ray' = spawn_ray(wo, ...)

        # Compute the directional radiance derivative
        J_ray', J_bsdf, J_Le = forward_grad(ray, {ray', bsdf_weight, L_e})
        J_bsdf = J_bsdf @ J_ray
        J_Le   = J_Le @ J_ray
        J_ray  = J_ray' @ J_ray
        J_L   += β * J_Le + L_e * J_β
        J_β    = bsdf_weight * J_β + β * J_bsdf
        β     *= bsdf_weight
    return L, J_L

def sample_path_adjoint(ray, L, J_L, δL):
    β = 1, J_ray = I_4
    for i in range(N):
        L -= β * L_e(...)
        ω', bsdf_value, bsdf_pdf = sample_bsdf(...)
        bsdf_weight = bsdf_value / bsdf_pdf
        ray' = spawn_ray(wo, ...)

        J_ray', J_bsdf, J_Le = forward_grad(ray, {ray', bsdf_weight, L_e})
        J_bsdf = J_bsdf @ J_ray
        J_Le = J_Le @ J_ray
        J_ray = J_ray' @ J_ray

        # Update the directional radiance derivative
        J_L -= L / bsdf_val * J_bsdf + β * J_Le
        J'_L  = J_L @ (J_ray)^{-1}

        # Backpropagate gradients of the current BSDF value
        δ_π += backward_grad(bsdf_weight, δL * L / bsdf_weight)
        # Backpropagate through shading frame and
        # BSDF sampling calculation
        δ_π += backward_grad(ray', δL @ J'_L)

        β *= bsdf_weight
    return δ_π
```

Listing 3. Pseudocode of our attached backpropagation algorithm.

Practical considerations. We always perform three rendering steps: the first computes an ordinary primal image and uses a pseudorandom number seeding scheme that de-correlates it from the subsequent two gradient-related evaluation steps. The need for this separation to avoid bias was noted by Azinović et al. [2019]. The primal image is then used to evaluate the objective function and compute its gradient, which produces the *adjoint image* (i.e. the derivative of the rendered image with respect to the optimization objective) that we then backpropagate from the sensor to objects with differentiable parameters. These two steps rely on the coupled pair of algorithms presented earlier, which are perfectly correlated in the sense that they visit the exact same sequence of path vertices.

The need to perform three instead of two rendering passes comes at an extra cost, which means that our linear-time approach may ultimately be slower than existing two-pass methods when the problem to be solved is sufficiently simple so that lack of linear computation time or constant memory usage are not bottlenecks.

(a) Reference (b) Ours (w/o regularization)

(c) Ours (d) Conv. AD

Fig. 4. We show the effect of our random regularization of the ray Jacobian. In this example, we computed the gradient of the surface normals of the ridges on the glass cups. Without stochastic regularization, the diffuse material of the wooden table produces singular Jacobians matrices, and the resulting gradients are mostly invalid (NaN, shown as white). With a small amount of regularization ($\lambda = 0.01$), our gradient estimates are very close to those computed using conventional automatic differentiation.

Stochastic regularization. The Jacobian matrix relating adjacent path segments is singular when the sampling strategy lacks a dependence on the previous vertex. This case for example arises following an interaction with a diffuse material, whose sampling strategy does not depend on the incident direction.

We use the following simple regularization scheme to avoid numerical failure during the inversion process: at each vertex, we add a small amount of statistically independent noise with zero mean, which does not bias the resulting gradients. In particular, we add a diagonal matrix $\lambda \cdot \mathbf{I}_4 \cdot \text{sign}\,(u - 1/2)$ to the ray Jacobian, where $u \sim \mathcal{U}(0, 1)$ is a uniform variate, $\mathbf{I}_4$ the 4×4 identity matrix and $\lambda = 0.01$ denotes a regularization weight. We again make sure to use the exact same random variates in both of our computation passes (i.e. the second and third passes according to the unbiased evaluation scheme discussed above). By using the *same* noise matrix in both passes, we ensure that our method remains unbiased. A detailed proof is provided in Section 2 of the supplemental material. The effect of this regularization is illustrated in Figure 4.

Moving discontinuities. Section 3 introduced the assumption that discontinuities, when present, do not depend on π and are thus *static* within the integration domain. A serious issue can arise when the π-dependent parameterization introduces an undesirable parameter dependence in previously static discontinuities. Without additional precautions, such a combination can then lead to in biased gradients.

The previously mentioned concurrent article [Zeltner et al. 2021] studies this situation and proposes a combination with an auxiliary transformation that "slows down" the parameterization near discontinuities to circumvent issues with bias. The auxiliary transformation builds on the frameworks of Loubet et al. [2019] and Bangaru et al. [2020] and requires a nested Monte Carlo evaluation of a convolution kernel that adds some implementation complexity and computational cost. In practice, bias due to naïve attached sampling may be acceptable given the simplicity and efficiency of the uncorrected approach. For example, Nimier-David et al. [2019, Section 4.3] demonstrate a working caustic design application that applies AD to an entire specular chain, which is functionally equivalent an attached sampler without corrections for moving discontinuities.

We refer to Zeltner et al. [2021] for details on discontinuity handling and will pay no further attention to this aspect, as our focus here is purely on algorithmic details like time and storage complexity of reverse-mode attached derivatives.

4.3 Mathematical interpretation: Iterative Jacobian Inversion

The previously discussed method performs an iterative reconstruction of the incident illumination along with its directional derivative in the attached case. We can further generalize this idea using an abstract view that interprets sampling of a light path as the repeated composition of a function h. In an implementation, h would represent the body of a `for` loop that takes the previous iteration's state as input to compute an updated set of state variables. In the basic path tracer from Listing 1, the relevant loop state is comprised of the value L and throughput β. The loop then evaluates

$$(L(\pi), \beta(\pi)) = \underbrace{h(\pi, h(\pi, ..., h(\pi, L_0, \beta_0))\cdots)}_{N} = h^{(N)}(\pi, L_0, \beta_0), \quad (11)$$

where $L_0 = 0$ and $\beta_0 = 1$. In this particular example, the function was defined as $h(L, \beta) = (L + \beta \cdot L_e(...), \beta \cdot f_s(...))$. For notational clarity, we omit the division of the BSDF value $f_s(...)$ by the sampling density. Using the chain rule, we can take the derivative of Equation (11) with respect to the scene parameter π:

$$\begin{aligned}\partial_\pi \left(L(\pi), \beta(\pi)\right) &= \partial_\pi \left[h(\pi, h(\pi, ..., h(\pi, L_0, \beta_0))\cdots)\right] \\ &= \sum_{k=1}^{N} \left[\prod_{j=k+1}^{N} J_h(L_j, \beta_j)\right] \partial_\pi h(\pi, L_{k-1}, \beta_{k-1}), \quad (12)\end{aligned}$$

where $(L_k, \beta_k) = h^{(k)}(\pi, L_0, \beta_0)$. This expression can be evaluated in different ways. Conventional AD techniques would store the function evaluations after each iteration of the forward loop and then compute the product of the Jacobians in reverse order. This is for example how the backpropagation algorithm for neural networks [Rumelhart et al. 1986] operates. Alternatively, we could run the computation in reverse and recompute quantities, which Gomez et al [2017] demonstrate on a reversible residual network.

We choose a different approach: instead of inverting the primal calculation, we invert the local Jacobian matrix relating the loop state of adjacent iterations. This is feasible, as the involved quantities are low-dimensional and the computation has a simple structure. The same approach would clearly not scale to neural networks with thousand-dimensional latent representations in intermediate layers.

The Jacobian J_h of the function h is given by

$$J_h = \begin{pmatrix} 1 & L_e(...) \\ 0 & f_s(...) \end{pmatrix}, \quad (13)$$

and we define $J_{h,k}$ as product of J_h over the path suffix at vertex k:

$$J_{h,k} = \prod_{j=k+1}^{N} J_h(L_j, \beta_j) = \begin{pmatrix} 1 & L_{k,N} \\ 0 & \beta_{k,N} \end{pmatrix}. \quad (14)$$

The subscript k, N denotes quantities that are accumulated from path vertex k to N. This shows that the indirect illumination term that needs to be evaluated during backpropagation can be interpreted as an entry of the Jacobian product. Plugging this back into the expression for $\partial_\pi L$ in Equation (12), we obtain

$$\begin{aligned}\partial_\pi \left(L(\pi), \beta(\pi)\right) &= \sum_{k=1}^{N} \begin{pmatrix} 1 & L_{k,N} \\ 0 & \beta_{k,N} \end{pmatrix} \partial_\pi h(\pi, L_{k-1}, \beta_{k-1}) \\ &= \sum_{k=1}^{N} \begin{pmatrix} 1 & L_{k,N} \\ 0 & \beta_{k,N} \end{pmatrix} \left[\begin{pmatrix} 0 \\ \beta_{k-1} \end{pmatrix} \partial_\pi f_s + \begin{pmatrix} \beta_{k-1} \\ 0 \end{pmatrix} \partial_\pi L_e \right] \\ &= \sum_{k=1}^{N} \beta_{k-1} \begin{pmatrix} 1 & L_{k,N} \\ 0 & \beta_{k,N} \end{pmatrix} \begin{pmatrix} \partial_\pi L_e \\ \partial_\pi f_s \end{pmatrix}. \quad (15)\end{aligned}$$

In practice, only the first component $\partial_\pi L$ of the gradient is desired, and we ignore the derivative of the path throughput β. Each iteration of the adjoint pass in Listing 2 then accumulates one of the elements of the previous sum into the scene parameter gradient. This equation now enables a new interpretation of the method presented in Section 4.1: the incident radiance computed in a first forward pass provides the Jacobian product $J_{h,0}$. As we now run our backward pass and subtract the current emitted radiance, we are effectively iteratively applying the inverse Jacobian matrix:

$$J_h^{-1} = \begin{pmatrix} 1 & -L_e(...)/f_s(...) \\ 0 & 1/f_s(...) \end{pmatrix}. \quad (16)$$

The throughput-related computation simplifies to a single division by the current BSDF value.

4.4 Differentiable Delta Tracking

Path replay backpropagation unlocks the door to efficient reverse-mode differentiation of volume transport based on unbiased *null-collision* methods like *delta tracking* [Woodcock et al. 1965]. Such methods alter the composition of the medium by introducing fictitious (i.e., optically inactive) matter that enables free-flight sampling via an unbiased rejection sampling process.

Differentiating such algorithms in reverse mode previously involved three separate sources of difficulty: first, light paths in participating media are generally much longer and can reach 100-1000s of scattering interactions, exacerbating the difficulty of program reversal. Second, null-scattering can introduce a large number of

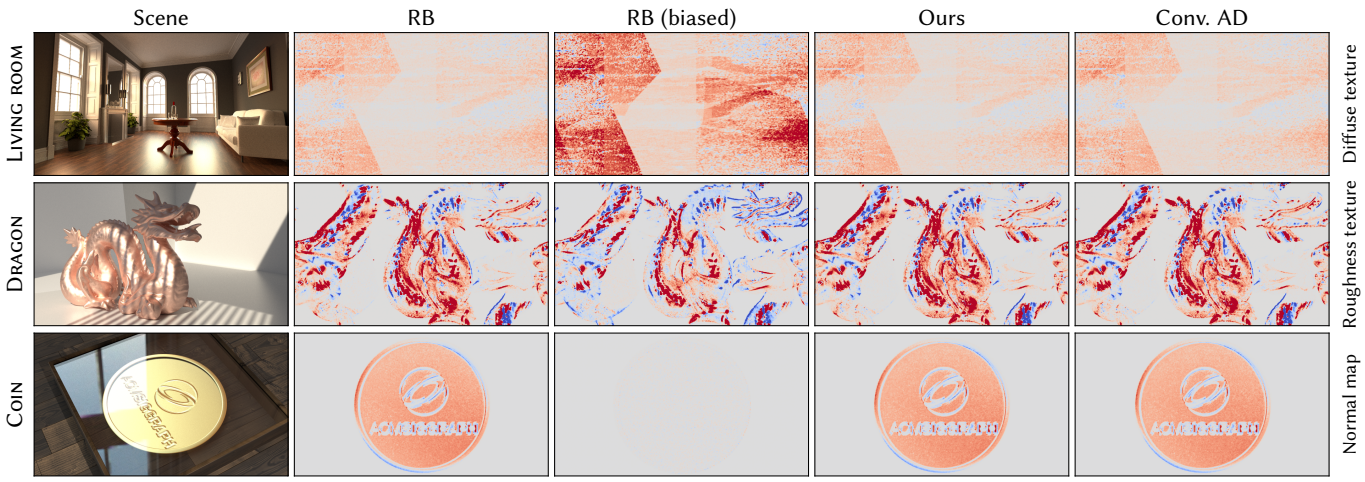


Fig. 5. Validation of gradients computed using *detached* sampling strategies: we visualize several types of parameter gradients in a somewhat construed validation testcase that involves backpropagating the difference to a blurred version of the primal image. In the first row, we compute the gradient of the diffuse albedo of the wooden floor. Only one dimension is shown in the case of multidimensional parameters like the albedo. The second row shows the gradient of a roughness texture on the dragon. Finally, the last row shows the gradient of the normal map of the embossed coin. Our method and the quadratic time version of radiative backpropagation match reference gradients obtained using conventional automatic differentiation. Biased RB can compute the gradient of a diffuse texture albeit with an incorrect scale, and it produces gradients of the wrong sign for both roughness and normal maps.

additional *null* interactions that expand the size of the intermediate program state even further. Third, null-scattering makes discrete decisions that require additional precautions during differentiation.

The former two issues are easily addressed by switching to detached or attached PRB with generalizations for volumes (e.g., scattering by a phase function in addition to the BSDF). To resolve the third issue, consider the null-scattering integral form of the radiative transfer equation [Galtier et al. 2013; Kutz et al. 2017]:

$$L_i(\mathbf{x}) = \int_0^\infty p(t) \left[\frac{\sigma_a(\mathbf{x}_t)}{\bar{\sigma}} L_e(\mathbf{x}_t) + \frac{\sigma_s(\mathbf{x}_t)}{\bar{\sigma}} L_s(\mathbf{x}_t) + \frac{\sigma_n(\mathbf{x}_t)}{\bar{\sigma}} L_i(\mathbf{x}_t) \right] \mathrm{d}t,$$

where we have omitted the dependence on ω for readability, $\bar{\sigma}$ is the majorant on extinction $\sigma_t = \sigma_a + \sigma_s$ evaluated at $\mathbf{x} + t\omega$, and σ_a, σ_s, σ_n denote the medium's absorption, scattering, and null scattering coefficient. The function $p(t) = \bar{\sigma} \exp(-t\bar{\sigma})$ is the free-flight density of the majorant that we assume to be a constant that does not participate in the differentiation process. Finally, L, L_e, and L_s are incident, emitted, and in-scattered radiance. We refer to Novak et al. [2018] for a thorough discussion of this formulation. To differentiate this integral, we use the following detached estimator:

$$\begin{aligned} \langle \partial_\pi L_i(\mathbf{x}) \rangle = {} & \partial_\pi \left[\psi_a(\mathbf{x}_t)\, L_e(\mathbf{x}_t) \right] \psi_a^{-1}(\mathbf{x}_t)\, \mathcal{H}[u < \psi_a(\mathbf{x}_t)] \\ & + \partial_\pi \left[\psi_s(\mathbf{x}_t)\, L_s(\mathbf{x}_t) \right] \psi_s^{-1}(\mathbf{x}_t)\, \mathcal{H}[\psi_a(\mathbf{x}_t) \le u < \psi_t(\mathbf{x}_t)] \\ & + \partial_\pi \left[\psi_n(\mathbf{x}_t)\, L_i(\mathbf{x}_t) \right] \psi_n^{-1}(\mathbf{x}_t)\, \mathcal{H}[\psi_t(\mathbf{x}_t) \le u], \end{aligned} \tag{17}$$

where the Heaviside function $\mathcal{H}$ equals 1 if the specified condition is satisfied and 0 otherwise. The variable t is sampled according to $p(t)$, $u \sim U(0,1)$, and $\psi_\circ(\mathbf{x}) := \sigma_\circ(\mathbf{x})/\bar{\sigma}$. Note that all functions $\psi_\circ$ and $L_\circ$ have an implicit dependence on π. The particle proportions $\psi_\circ$ occur twice in each row, but they must only differentiated once as derivatives involving the discrete choice between absorption and (null-)scattering would severely bias the resulting estimator. Zeltner et al. [2021, Section 3.2] discuss the reasons for this in further detail.

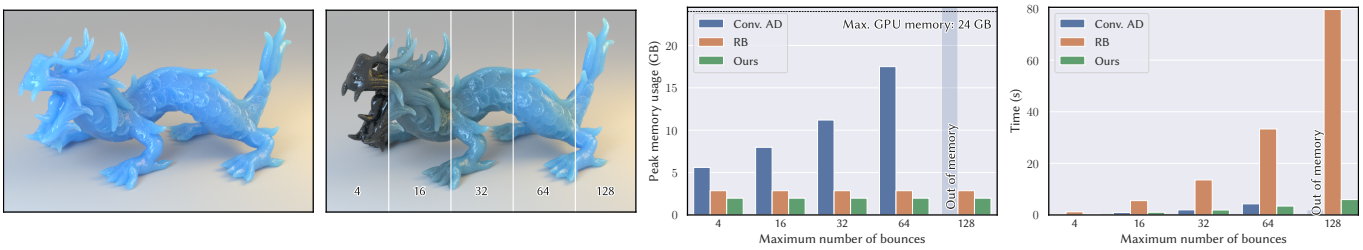
Practical implementations of this method will likely also build on specialized direct illumination sampling strategies referred to as *next event estimation*. One additional challenge that arises here is that transmittance evaluation towards the sampled light position using a method like *ratio tracking* [Novák et al. 2014] adds another recursive loop over an unbounded number of scattering events. To correctly differentiate this process, we rely on a second nested application of PRB within the volumetric path tracer loop. Section 3 of the supplemental material provides further details and pseudocode.

We have not yet experimented with *attached* versions of volume transport estimators but consider them an interesting avenue for future work, especially to handle the directional domain of the in-scattering integral: light paths in forward-scattering media ($g > .99$) are highly constrained and reminiscent of specular chains in the surface case. Simple detached sampling strategies can be a poor choice for such a directionally peaked integrand [Zeltner et al. 2021].

5 RESULTS

We turn to results and present correctness tests, several applications, and performance evaluations. We evaluated our implementation in a differentiable rendering system based on Mitsuba 2 [Nimier-David et al. 2019] on a Intel i7-7800x workstation with a NVIDIA TITAN RTX graphics card (23 GiB of RAM) and using OptiX 7.2 [Parker et al. 2010] for hardware-accelerated ray tracing.

Gradients from detached sampling strategies. In Figure 5, we validate the correctness of our method by comparing gradients computed using detached sampling strategies to conventional automatic differentiation and both biased and unbiased RB variants. This figure provides further demonstration that biased RB generates gradients with an incorrect sign. This effect is particularly pronounced when the directional distribution of incident radiance plays a strong role, for instance when optimizing normal or roughness maps.



(a) Scene (b) Appearance with N bounces (c) Memory usage (d) Gradient render time

Fig. 6. In this example, we compute gradients with respect to the dragon's subsurface scattering albedo. This represents a typical case where long paths with over a hundred scattering events must be simulated to faithfully capture the material appearance. In (b), we show how the appearance would degrade if we excluded higher order bounces. We then compute the gradient of the albedo using conventional automatic differentiation, unbiased radiative backpropagation (RB), and our new unbiased method. Despite rendering the scene only at 640 × 360 pixels and 1 sample per pixel, conventional AD quickly exhausts the available memory of a TITAN RTX GPU. Runtime of unbiased RB grows quadratically with path length and becomes prohibitively slow when many scattering events are considered. In contrast, our method performs unbiased estimates using a constant memory footprint and only a linear increase in computation time.

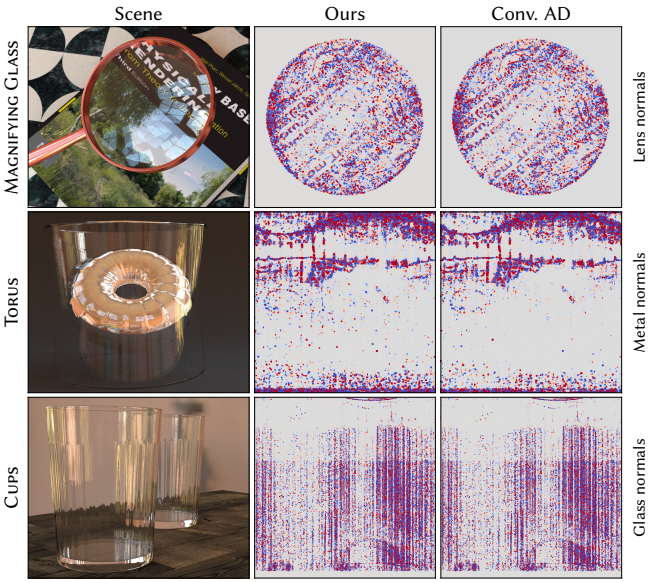


Fig. 7. Similar to the detached case in Figure 5, we now visualize the gradients of *attached* sampling strategies that track differential changes of the path geometry with respect to perturbations of the scene parameters. Parameter optimization of perfectly specular objects requires this approach. The example in the first row shows a magnifying glass with curvature modeled using a normal map. The second scene has normal map on an indirectly observed metallic torus, and the last row uses a normal map to model the profile of the two glass cups.

Gradients from attached sampling strategies. Conversely, Figure 7 shows gradients produced by *attached* sampling strategies. The normal map gradients were computed using paths of length 12 using 128 samples per pixel. Specular interreflection along with both reflection and refraction makes it possible to reach each texture-position using various different path configurations. This leads to a relatively high amount of variance compared to the previous case. Radiative backpropagation does not support this type of computation and will return a zero-valued gradient in all three cases.

Subsurface scattering. Our method outperforms prior work in scenes that are characterized by light paths with many scattering events. Figure 6 showcases the asymptotic behavior of gradient evaluation with respect to the subsurface scattering albedo of a dielectric object with homogeneous internal scattering. The high average path length makes both recursive radiative backpropagation and conventional automatic differentiation approaches unsuitable: the former suffers from quadratic computational cost to recursively estimate the incident illumination, while the latter requires storage of intermediate program state for differentiation that grows linearly with path length and quickly exhausts all available GPU memory.

Heterogeneous volume optimization. In Figure 9, we use our method to optimize a heterogeneous volume using delta tracking. The large number of null scattering events makes any method with a super-linear dependence on path length impractical. Delta tracking maps extremely poorly to wavefront style rendering, so conventional AD, which runs a big wavefront of rays, is unable to complete even a single iteration. Figure 1 showcases a similar albedo and density optimization under difficult illumination conditions.

Performance evaluation. We evaluate the computation time and memory requirements of both our detached and attached derivatives in Figure 8. The results confirm that our method and radiative backpropagation both use a constant amount of memory, which can be substantially lower than memory requirements of conventional AD. While we build on a relatively optimized AD implementation, this approach still uses in excess of 10 GiB of memory to compute a small number of gradient samples in several simple tests.

Optimization using attached sampling. Figure 10 shows how attached PRB can be used to optimize the normal map of a glass slab. Our method matches the results achieved using conventional AD.

6 CONCLUSION AND FUTURE WORK

We presented a new linear-time and constant-memory approach to differentiate the image formation process of physically-based rendering. Our method has a cost that is similar to a biased method presented in previous work, but its unbiased nature makes it more reliable in many types of simulations.

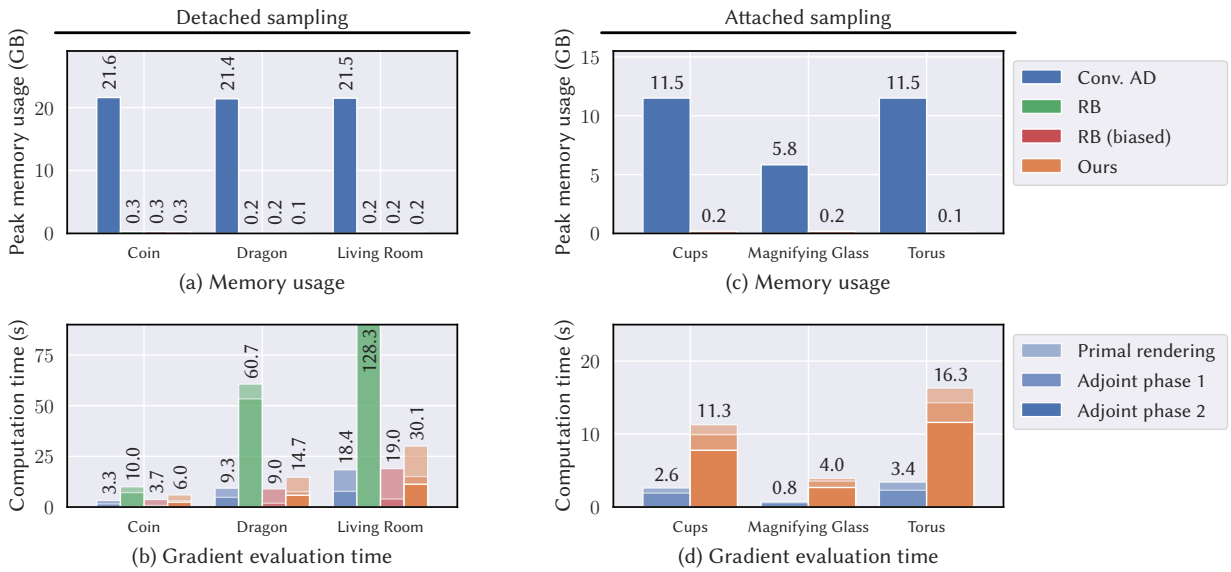
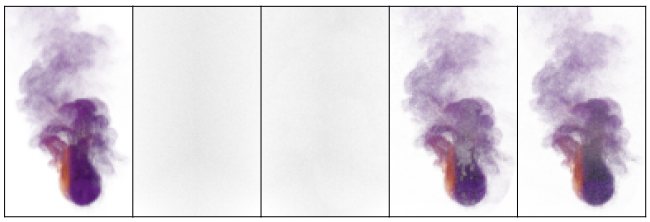


Fig. 8. We compare the runtime and memory usage of differentiable rendering in multiple example scenes, separating the time spent in the primal and two differentiation-related passes. The primal pass uses a larger number (4 ×) of samples compared to the adjoint passes, which has a positive effect on gradient variance [Azinović et al. 2019]. In our method, the first adjoint pass refers to the precomputation of temporary information (radiance estimate, ray Jacobians) consumed by the final adjoint pass that accumulates scene parameter derivatives.



	(a) Reference	(b) Conv. AD	(c) RB	(d) RB (biased)	(e) Ours
Time:		45 min	3 min	1 min 47 s	1 min 32 s
Iterations:		0	4	96	64

Fig. 9. We optimize the density of a heterogeneous volume using delta tracking and multiple methods at roughly equal time. The number of sampled medium interactions makes both conventional automatic differentiation (conv. AD) and unbiased radiative backpropagation (RB) completely infeasible. Conventional AD fails to complete even one iteration within a timespan of 45 minutes. Our method and biased RB are both significantly faster. Our method seems converge slightly more reliably than biased RB.

We also show that this approach generalizes to the more complex case of attached sampling strategies that track the differential dependence of Monte Carlo importance sampling on scene parameters, enabling differentiation of degenerate BSDFs containing Dirac delta functions. Both methods use a constant amount of memory, which we expect to be crucial when optimizing large scenes. We also expand on previous work targeting volumetric appearance reconstruction and are the first to solve this problem using unbiased delta tracking. The advantages over ray marching match those observed in primal rendering, and we hope that this possibility will inspire inverse rendering to similarly shift towards unbiased volume rendering methods.

ACKNOWLEDGMENTS

We thank Tizian Zeltner for many useful discussions about attached derivatives and Baptiste Nicolet for his help drawing some of the figures. We used several meshes from the Stanford 3D scanning repository. The Living Room scene has been created by Wig42 on Blendswap.com and converted to the Mitsuba format by Benedikt Bitterli. This research was supported by the Swiss National Science Foundation (SNSF) as part of grant 200021_184629.

REFERENCES

Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. 2019. Inverse Path Tracing for Joint Material and Lighting Estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE.*

Sai Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020), 245:1–245:18.

Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Towards Learning-based Inverse Subsurface Scattering. In *2020 IEEE International Conference on Computational Photography (ICCP).* IEEE, 1–12.

Min Chen and James Arvo. 2000. Theory and Application of Specular Path Perturbation. *ACM Trans. Graph.* 19, 4 (Oct. 2000), 246–278.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems.* 6571–6583.

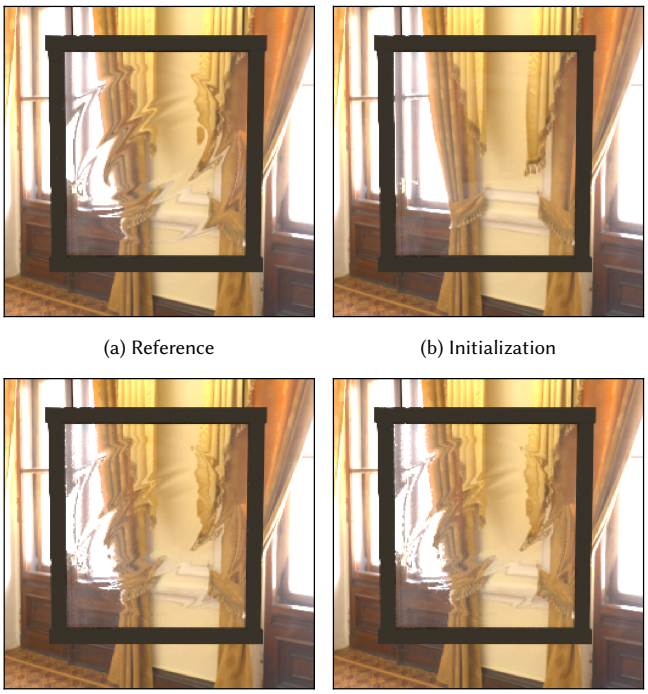
Luc Devroye. 1986. *Non-Uniform Random Variate Generation.* Springer-Verlag.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. NICE: Non-linear Independent Components Estimation. In *3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, Workshop Track Proceedings.* arXiv:1410.8516

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using Real NVP. arXiv:1605.08803

Mathieu Galtier, Stéphane Blanco, Cyril Caliot, Christophe Coustet, Jérémi Dauchet, Mouna El Hafi, Vincent Eymet, Richard Fournier, Jacques Gautrais, Anaïs Khuong, et al. 2013. Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer* 125 (2013).

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision.* Springer, 685–701.



(a) Reference (b) Initialization

(c) Conv. AD (d) Ours

Fig. 10. We optimize the normal map of a glass slab so that the refracted view matches a reference image. Neither biased nor unbiased radiative backpropagation can optimize a perfectly specular surface, hence we compare our method to conventional automatic differentiation (at equal iteration count).

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse Volume Rendering with Material Dictionaries. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 32, 6, Article 162 (Nov. 2013).

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. 2017. The Reversible Residual Network: Backpropagation without Storing Activations. In *NIPS*.

Andreas Griewank and Andrea Walther. 2000. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)* 26, 1 (2000), 19–45.

Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Vol. 105. SIAM.

Laurent Hascoet and Valérie Pascual. 2013. The Tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software (TOMS)* 39, 3 (2013), 1–43.

Carole K. Hayakawa, Jerome Spanier, Frédéric Bevilacqua, Andrew K. Dunn, Joon S. You, Bruce J. Tromberg, and Vasan Venugopalan. 2001. Perturbation Monte Carlo methods to solve inverse photon migration problems in heterogeneous tissues. *Opt. Lett.* 26, 17 (Sep 2001), 1335–1337.

Shayan Hoshyari, Hongyi Xu, Espen Knoop, Stelian Coros, and Moritz Bächer. 2019. Vibration-minimizing motion retargeting for robotic characters. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 2019. A Survey on Gradient-Domain Rendering. *Computer Graphics Forum* 38, 2 (2019), 455–472.

Homan Igehy. 1999. Tracing Ray Differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 99)*. ACM Press/Addison-Wesley Publishing Co., USA, 179–186.

Wenzel Jakob and Steve Marschner. 2012. Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport. *ACM Trans. Graph.* 31, 4 (July 2012).

James T. Kajiya. 1986. The Rendering Equation. In *Proc. of Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. ACM, New York, NY, USA, 143–150.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2015. Matching Real Fabrics with Micro-Appearance Models. *ACM Trans. Graph.* 35, 1, Article 1 (Dec. 2015), 26 pages.

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4, Article 111 (2017), 111:1–111:16 pages.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (2018), 222:1–222:11.

Seppo Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics* 16, 2 (1976), 146–160.

Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. 2019. Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction. *CoRR* (2019). arXiv:1901.05567

Matthew M Loper and Michael J Black. 2014. OpenDR: An approximate differentiable renderer. In *European Conference on Computer Vision*. Springer.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Transactions on Graphics* 38, 6 (Dec. 2019).

Iván Lux and Lázló Koblinger. 1990. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC Press, Boston.

Jiahui Lyu, Bojian Wu, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2020. Differentiable Refraction-Tracing for Mesh Reconstruction of Transparent Objects. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020), 195:1–195:13.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. In *ACM Transactions On Graphics (TOG)*, Vol. 23. ACM, 449–456.

Don Mitchell and Pat Hanrahan. 1992. Illumination from curved reflectors. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 283–291.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4, Article 146 (July 2020), 15 pages.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (Nov. 2019), 17 pages.

Jan Novák, Iliyan Georgiev, Johannes Hanika, Jaroslav Křivánek, and Wojciech Jarosz. 2018. Monte Carlo Methods for Physically Based Volume Rendering. In *ACM SIGGRAPH 2018 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. ACM, New York, NY, USA, 14:1–14:1.

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 33, 6 (Nov. 2014).

Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph. (Proc. SIGGRAPH)*, Article 66 (2010), 13 pages.

Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. 2019. Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer. *CoRR* (2019). arXiv:1903.11149

Lev Semenovich Pontryagin. 1962. *Mathematical theory of optimal processes*. CRC Press.

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Transactions on Graphics (TOG)* 26, 1 (2007).

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A Versatile Scene Model with Differentiable Visibility Applied to Generative Pose Estimation. In *Proceedings of ICCV 2015*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning Representations by Back-propagating Errors. *Nature* 323, 6088 (1986), 533–536. http://www.nature.com/articles/323533a0

NH Shimada and Toshiya Hachisuka. 2020. Quantum Coin Method for Numerical Integration. In *Computer Graphics Forum*. Wiley Online Library.

Jeffrey Mark Siskind and Barak A. Pearlmutter. 2008. Nesting forward-mode AD in a functional framework. *Higher-Order and Symbolic Computation* 21, 4 (2008), 361–376.

JM Tregan, S Blanco, J Dauchet, M Hafi, R Fournier, L Ibarrart, P Lapeyre, and N Villefranque. 2019. Convergence issues in derivatives of Monte Carlo null-collision integral formulations: a solution. (2019). arXiv:1903.06508

Eric Veach and Leonidas Guibas. 1995. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.

Greg Ward and Paul Heckbert. 1992. *Irradiance gradients.* Technical Report. Lawrence Berkeley Lab., CA (United States); Ecole Polytechnique Federale, Lausanne (Switzerland); Technische Hogeschool Delft (Netherlands). Dept. of Technical Mathematics and Informatics.

E. R. Woodcock, T. Murphy, P. J. Hemmings, and T. C. Longworth. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. *Applications of Computing Methods to Reactor Problems* (1965).

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020).

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo Estimators for Differential Light Transport. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4 (Aug. 2021).

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4, Article 143 (July 2020), 19 pages.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6, Article 227 (Nov. 2019), 16 pages.

Shaung Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling Scattering Parameters for Rendering Anisotropic Media. *ACM Transactions on Graphics* 35, 6 (2016).