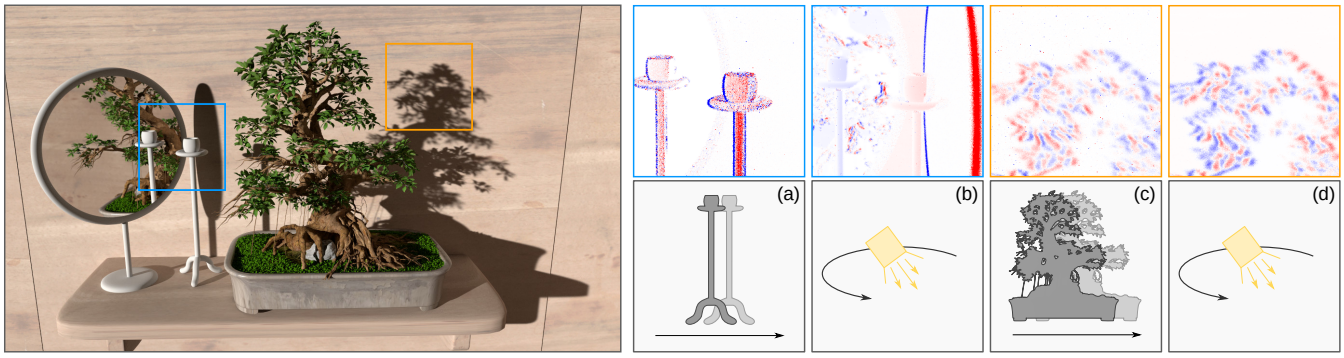
Reparameterizing Discontinuous Integrands for Differentiable Rendering

Guillaume Loubet, Nicolas Holzschuch, Wenzel Jakob

Reparameterizing Discontinuous Integrands for Differentiable Rendering

GUILLAUME LOUBET, École Polytechnique Fédérale de Lausanne (EPFL)
NICOLAS HOLZSCHUCH, Inria, Univ. Grenoble-Alpes, CNRS, LJK
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL)



A scene with complex geometry and visibility (1.8M triangles)

Gradients with respect to scene parameters that affect visibility

Fig. 1. The solution of inverse rendering problems using gradient-based optimization requires estimates of pixel derivatives with respect to arbitrary scene parameters. We focus on the problem of computing such derivatives for parameters that affect visibility, such as the position and shape of scene geometry (a, c) and light sources (b, d). Our renderer re-parameterizes integrals so that their gradients can be estimated using standard Monte Carlo integration and automatic differentiation—even when visibility changes would normally make the integrands non-differentiable. Our technique produces high-quality gradients at low sample counts (64 *spp* in these examples) for changes in both direct and indirect visibility, such as glossy reflections (a, b) and shadows (c, d).

Differentiable rendering has recently opened the door to a number of challenging inverse problems involving photorealistic images, such as computational material design and scattering-aware reconstruction of geometry and materials from photographs. Differentiable rendering algorithms strive to estimate partial derivatives of pixels in a rendered image with respect to scene parameters, which is difficult because visibility changes are inherently non-differentiable.

We propose a new technique for differentiating path-traced images with respect to scene parameters that affect visibility, including the position of cameras, light sources, and vertices in triangle meshes. Our algorithm computes the gradients of illumination integrals by applying changes of variables that remove or strongly reduce the dependence of the position of discontinuities on differentiable scene parameters. The underlying parameterization is created on the fly for each integral and enables accurate gradient estimates using standard Monte Carlo sampling in conjunction with automatic differentiation. Importantly, our approach does not rely on sampling silhouette edges, which has been a bottleneck in previous work and tends to produce high-variance gradients when important edges are found with insufficient probability in scenes with complex visibility and high-resolution geometry. We show that our method only requires a few samples to produce gradients with low bias and variance for challenging cases such as glossy reflections and shadows. Finally, we use our differentiable path tracer to reconstruct the 3D geometry and materials of several real-world objects from a set of reference photographs.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Ray tracing**.

Additional Key Words and Phrases: differentiable rendering, inverse rendering, stochastic gradient descent, discontinuous integrands, path tracing

ACM Reference Format:
Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.* 38, 6, Article 228 (November 2019), 14 pages. https://doi.org/10.1145/3355089.3356510

1 INTRODUCTION

Physically based rendering algorithms generate photorealistic images by simulating the flow of light through a detailed mathematical representation of a virtual scene. Historically a one-way transformation from scene to rendered image, the emergence of a new class of differentiable rendering algorithms has enabled the use of rendering in an inverse sense, to find a scene that maximizes a user-specified objective function. One particular choice of objective leads to *inverse rendering*, whose goal is the acquisition of 3D shape and material properties from photographs of real-world objects, alleviating the tedious task of modeling photorealistic content by hand. Other kinds of objective functions hold significant untapped potential in areas like computational material design and architecture: for instance, inverse rendering could be used to create pigment

Authors' addresses: Guillaume Loubet, École Polytechnique Fédérale de Lausanne (EPFL), g.loubet.research@gmail.com; Nicolas Holzschuch, Inria, Univ. Grenoble-Alpes, CNRS, LJK, nicolas.holzschuch@inria.fr; Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL), wenzel.jakob@epfl.ch.

0730-0301/2019/11-ART228 $15.00
https://doi.org/10.1145/3355089.3356510

mixtures [Papas et al. 2013], refractive caustics [Schwartzburg et al. 2014], or help optimize daylighting as part of an architectural design process [Andersen et al. 2008].

The first differentiable renderers [Patow and Pueyo 2003] were able to compute gradients with respect to simple material parameters (e.g. scattering albedo or roughness) but did not handle more challenging parameters that affect the shape and placement of objects in a rendered image (e.g. vertex positions and camera pose). The latter are related to non-differentiable visibility terms in illumination integrands [Kajiya 1986] that prevent the standard approach of differentiating under the integral sign. Neglecting their influence in turn leads to incorrect gradient estimates that are unusable for optimization. Later work has focused on differentiating the parameters of volumetric transport simulations [Gkioulekas et al. 2013; Khungurn et al. 2015; Zhao et al. 2016] and approximately differentiable rasterization of meshes or volumes, ignoring global light transport effects [Loper and Black 2014; Rhodin et al. 2015].

Recently, Li et al. [2018] presented the first differentiable rendering technique that simultaneously accounts for higher-order transport phenomena and visibility-related discontinuities. Their technique augments path tracing with an additional step that samples positions on *silhouette edges* with respect to a given scene position: the camera position in the case of *primary visibility*, and arbitrary scene positions in the case of *indirect visibility*. While the former set of edges can be found in a brief preprocess step, indirect visibility poses a serious challenge because the set of silhouette edges depends on the point being shaded. Li et al. construct a 6D data structure in 3D Hough space [Olson and Zhang 2006] to sample suitable edges, but this strategy tends to find visible edges with insufficient density, producing gradients with high variance. This issue grows in severity as the geometric complexity of objects in the scene increases.

Our main contribution is a new technique for differentiable path tracing that addresses this limitation. We observe that explicit sampling of discontinuities can be avoided by applying carefully chosen changes of variables that remove the dependence of discontinuities on scene parameters. The resulting integrands of course still contain discontinuities, but they are *static* in the re-parameterized integral and hence no longer prevent differentiation under the integral sign. Following this idea, we propose a differentiable renderer that applies a suitable change of variables to each integral using an approximation of the ideal parameterization that is simple to compute using ordinary ray tracing operations. Our method does not require edge sampling and evaluates illumination integrals using standard Monte Carlo sampling in conjunction with automatic differentiation. We furthermore propose a variance reduction technique for gradients using control variates with pairs of correlated paths.

We demonstrate that our technique estimates gradients with low bias and variance even when few samples are taken and gradients arise due to difficult indirect visibility, such as shadows and glossy reflections, and that it outperforms previous work in such cases. Unlike silhouette edge sampling, our technique produces high-quality gradients in scenes with very high geometric complexity. Finally, we apply our method to a challenging inverse rendering problem to reconstruct the 3D geometry and material parameters of real-world objects from a set of reference photographs. The open source implementation of our method is part of version 2 of the Mitsuba renderer available at https://mitsuba-renderer.org.

2 RELATED WORK

2.1 Automatic differentiation

A differentiable path tracer is able to propagate derivative information through a complex simulation that involves the interaction of many different system components including light sources, BSDFs, textures, cameras, and so on. Analytic derivatives can be determined by hand for each component, but this is tedious and error-prone. Finite-difference techniques are simpler to use but do not scale to high-dimensional parameter spaces.

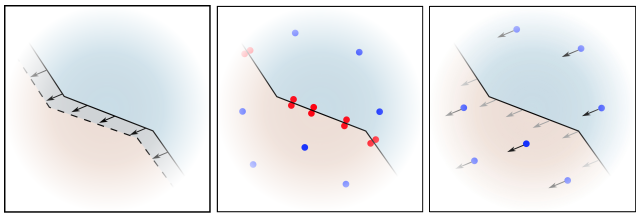
Automatic Differentiation (AD) provides a powerful tool to automate the computation of derivatives via systematic application of the chain rule [Griewank and Walther 2008; Wengert 1964]. Our implementation relies on a particular variant known as *reverse-mode AD*, which is efficient when there are many inputs (e.g. the vertex positions of a high-resolution mesh) and an objective function that quantifies the quality of a solution ("loss"). Reverse-mode AD is also known as *backpropagation* in the context of neural networks [Rumelhart et al. 1986] and has been used by previous differentiable renderers [Che et al. 2018]. We note that the core contribution of our work is largely independent of the specifics of how derivatives are computed.

2.2 Differentiating continuous illumination integrals

Inverse problems involving multiple scattering in participating media are notoriously challenging due to the global and highly nonlinear relationship between scattering parameters and pixel values. Gkioulekas et al. [2013] differentiate a dictionary of reference materials and apply stochastic gradient descent to determine scattering parameters matching a measured translucent material. Hašan and Ramamoorthi [2013] showed that analytic differentiation of pixel values leads to nested integrals that resemble light transport integrals, allowing simultaneous estimation of gradients using standard path tracing. Several other differentiable volume renderers are similarly based on analytic differentiation of path contributions [Gkioulekas et al. 2016; Khungurn et al. 2015; Zhao et al. 2016]. Velinov et al. [2018] bypassed the problem of non-differentiable boundaries between media by defining a continuously varying media and optimizing the shape of the boundary. Various differentiable renderers have been proposed recently based on the same differentiability assumptions, relying on either analytic derivatives [**???**] or automatic differentiation [Che et al. 2018]. Our work is related to these techniques but specifically targets integrals with discontinuous integrands.

2.3 Differentiating with respect to shapes

The problem of finding parameters that explain an image has been studied in depth in computer vision and it is often referred to as *analysis by synthesis* or *inverse graphics*. The need for differentiable renderers for solving pose or shape estimation problems has been identified by a large number of authors that approach the problem in different ways, for instance by computing the occupancy



(a) Integrand with discontinuity (b) Edge sampling [Li et al. 2018] (c) Using changes of variables (ours)

Fig. 2. (a): Integrands in physically based rendering have discontinuities whose location depends on scene parameters such as object positions. (b): In previous work [Li et al. 2018], the gradients of these integrals were estimated by sampling pairs of paths around the important silhouette edges in each integrand (red dots) in addition to standard Monte Carlo samples (blue dots). (c): We bypass the costly task of sampling visible silhouette edges by using changes of variables such that the discontinuities do not move with respect to the Monte Carlo samples for infinitesimal changes of scene parameters. This leads to Monte Carlo estimates that can be differentiated using ordinary techniques, for instance using automatic differentiation.

of pixels [Jalobeanu et al. 2004; Smelyansky et al. 2002], identifying boundary pixels in image space [de La Gorce et al. 2008; Loper and Black 2014], rasterizing smooth triangles [Kato et al. 2018; Liu et al. 2019; Petersen et al. 2019] or volumes [Nguyen-Phuoc et al. 2018; Rhodin et al. 2015], and by deriving gradients for visibility-aware shape optimization [Delaunoy and Prados 2011; ?]. These techniques all focus on discontinuities that are directly observed by the camera, relying on simple direct lighting models. They do not account for gradients due to indirect modes of transport, such as shadows, interreflection or refraction. These additional cues can be used to guide the reconstruction of an object from reference photos by reducing ambiguities about its shape. Ramamoorthi et al. [?] derived image-space gradients for soft shadows, whose evaluation requires finding the silhouette edges that occlude the light sources at each shading point.

Recently, Li et al. introduced the first differentiable path tracer that provides gradients with respect to arbitrary scene parameters including the shape of visible objects and placement of light sources and camera [Li et al. 2018]. Similar to prior work, Li et al. differentiate illumination integrals and evaluate them using Monte Carlo sampling [Che et al. 2018; Gkioulekas et al. 2016; Khungurn et al. 2015; Velinov et al. 2018]. In contrast to prior work, they account for the effect of visibility changes by introducing an additional step that samples discontinuities in each integrand. This entails estimating the change in radiance perpendicular to an edge using a pair of nested estimates, as illustrated in Figure 2b. At the core of their algorithm is a technique for sampling visible edges in each integrand, which is also responsible for the main limitations of their approach. Indeed, finding silhouette edges that are visible from arbitrary points of the scene at run time is a problem that does not scale well with the complexity of the scene. The search for edges can be performed efficiently in Hough space [Olson and Zhang 2006] but their visibility from a given point cannot be precomputed, leading either to high variance or inefficient sample generation if the visibility of each edge is tested in scenes with many edges and complex visibility (Figure 1). Another issue is that only a small part of an edge may contribute due to weighting by other factors in the illumination integral, such as the associated BSDF model. Li et al. thus propose a technique for sampling position along edges, which requires conversion of all reflectance models into the Linearly Transformed Cosine representation [Heitz et al. 2016a]. Like them, our method estimates the gradients with respect to arbitrary parameters, but it neither requires sampling edges nor points on edges, allowing the use of arbitrary BSDF models without conversion to other representations. We compare both methods and demonstrate the superior robustness of our approach in complex scenes.

2.4 Variance reduction for Monte Carlo estimators

Well-known variance reduction techniques include control variates, antithetic variances, the reuse of random numbers, stratified sampling, as well as importance and multiple importance sampling [Veach 1998] that are essential to any kind of physically based rendering. Variance reduction techniques can be particularly effective when applied to Monte Carlo estimators of gradients. For instance, gradient-domain rendering techniques rely on shift maps [Kettunen et al. 2015; Lehtinen et al. 2013] to dramatically reduce variance by sampling pairs of correlated paths. Rousselle et al. [2016] apply control variates to reduce the variance of horizontal and vertical gradient estimates in an image. Our approach is related to both of these methods: we reduce variance of gradient estimates by tracing pairs of correlated paths that reuse certain random numbers, combining their estimates using control variates.

3 PRELIMINARIES

Physically based rendering relies on Monte Carlo estimators of pixel and shading integrals [Kajiya 1986], generally of the form

$$I = \int_{\mathcal{X}} f(x, \Theta)\, \mathrm{d}x, \tag{1}$$

where f is defined on a domain $\mathcal{X}$ (often the unit sphere S^2). The function f depends on scene parameters Θ, such as vertex positions, normals, textures, etc. This section explains how gradients of such integrals can be computed, and how a change of variables can facilitate this in the case of a non-differentiable integrand. To simplify notation, we assume without loss of generality that there is only a single scene parameter $\theta \in \mathbb{R}$.

3.1 Differentiating Monte Carlo estimators (smooth case)

Gradient-based optimization techniques require access to partial derivatives of the integral I with respect to θ:

$$\frac{\partial I}{\partial \theta} = \frac{\partial}{\partial \theta} \int f(x, \theta)\, \mathrm{d}x. \tag{2}$$

The Leibniz integral rule states that the existence and continuity of both f and its partial derivative in θ are sufficient conditions for differentiating under the integral sign, in which case

$$\frac{\partial}{\partial \theta} \int f(x, \theta)\, \mathrm{d}x = \int \frac{\partial}{\partial \theta} f(x, \theta)\, \mathrm{d}x. \tag{3}$$

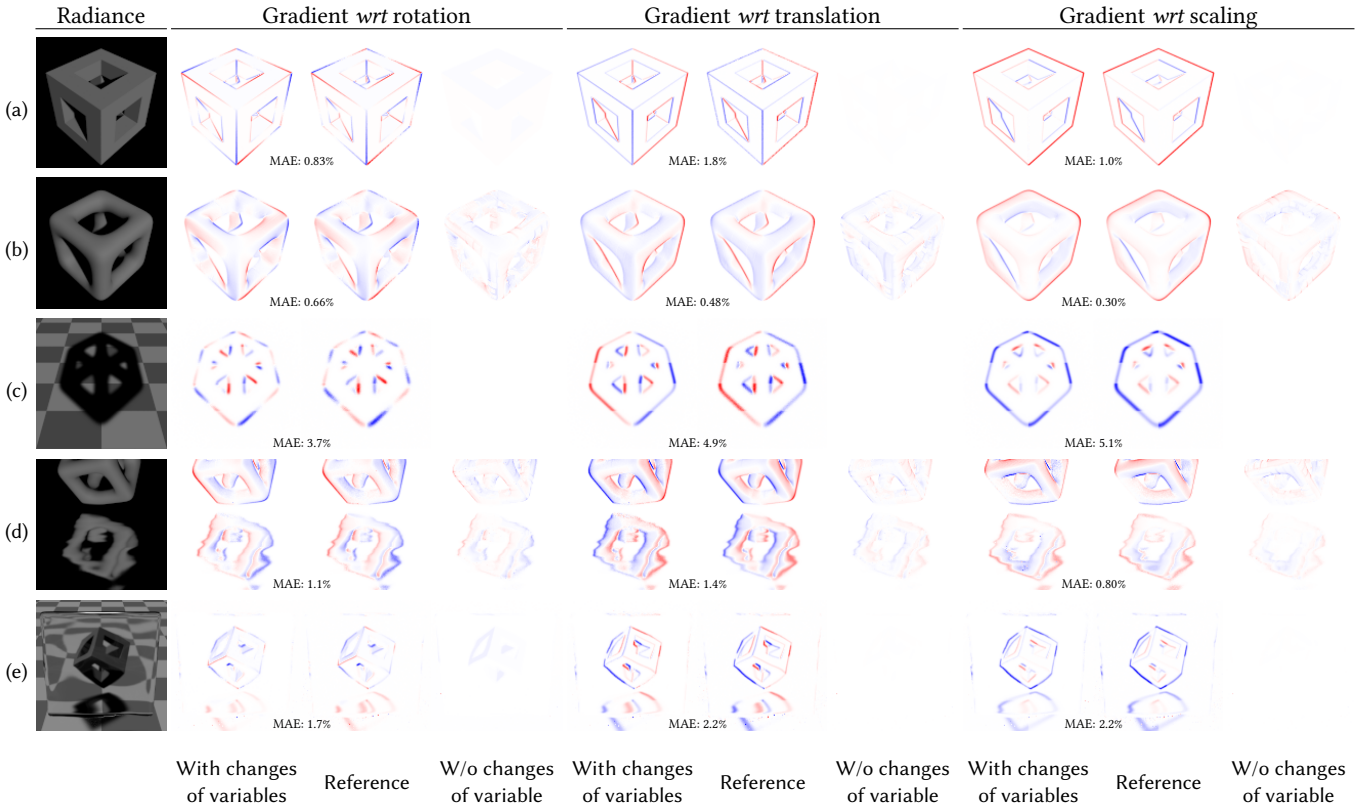


Fig. 3. Our method is able to compute accurate gradients with respect to scene parameters that affect geometry and visibility (e.g. object rotation, translation and scaling). In each case, we compare the results obtained using the change of variables formulation proposed in this work to naïve gradients and a reference obtained via finite differences. The rows of this figure show **(a)** An object with sharp features that lead to both visibility and shading discontinuities. **(b)** An object with the same topology as (a) but with a smooth surface. **(c)** The shadow of object (b) projected on a diffuse plane. **(d)** A reflection of object (b) from a rough metallic surface. **(e)** The object (a) seen through a rough dielectric slab. The mean absolute errors (MAE) are computed from the subset of pixels with nonzero gradients and are specified relative to the maximum gradient value observed in the corresponding reference. These errors reflect the bias of our estimators but also the residual noise in the images, in particular in examples with low MAEs such as (b).

When the integral I is estimated using Monte Carlo integration and a probability density function $p(x)$ that does *not* depend on θ, i.e.,

$$I \approx E = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i,\theta)}{p(x_i)}, \tag{4}$$

an estimator of the derivative is simply the derivative of the estimator:

$$\frac{\partial I}{\partial \theta} \approx \frac{1}{N}\sum\frac{\partial}{\partial\theta}\frac{f(x_i,\theta)}{p(x_i)} = \frac{\partial E}{\partial\theta}. \tag{5}$$

However, it is not uncommon that sampling techniques depend on a scene parameter θ, particularly when it has a significant effect on the shape of the integrand. In this case, Eq. 4 turns into

$$I \approx E = \frac{1}{N}\sum\frac{f(x_i(\theta),\theta)}{p(x_i(\theta),\theta)}. \tag{6}$$

When p is differentiable with respect to θ, and when the sampling procedure implements a differentiable mapping from random numbers to samples $x_i(\theta)$, then it is correct to differentiate Monte Carlo estimates in the same manner:

$$\frac{\partial I}{\partial \theta} \approx \frac{\partial E}{\partial \theta} = \frac{1}{N}\sum\frac{\partial}{\partial\theta}\frac{f(x_i(\theta),\theta)}{p(x_i(\theta),\theta)}. \tag{7}$$

This can be verified by re-parameterizing the integral I on the domain $\mathcal{U}$ of uniform input samples (also known as *primary sample space* [Kelemen et al. 2002]):

$$I = \int_{\mathcal{X}} f(x,\theta)\,\mathrm{d}x = \int_{\mathcal{U}}\frac{f(s(u,\theta),\theta)}{p(s(u,\theta),\theta)}\,\mathrm{d}u, \tag{8}$$

where the factor $1/p(s(u,\theta),\theta)$ in the denominator is the determinant of the Jacobian of the mapping. If p and s are differentiable, differentiation of Eq. 8 under the integral sign is valid, and uniform Monte Carlo sampling of $\mathcal{U}$ yields the gradient estimator in Eq. 7. Note that accounting for the dependence of p on θ generally yields higher-quality gradient estimates compared to the naïve estimator in Eq. 5, since many terms in Eq. 7 will cancel when the expression is differentiated.

Most mappings used in rendering systems are fortunately differentiable, but some notable exceptions do exist (e.g. sampling of multi-lobed BSDFs). A simple example of this problem and one possible solution can be found in Appendix A.

3.2 The case of non-differentiable integrands

Even when the mappings used for importance sampling are differentiable, it is often the case that the original integrand $f(x, \theta)$ is non-differentiable in θ due to visibility changes. These manifest as discontinuities, whose position in $\mathcal{X}$ are a function of θ. In such cases, differentiation under the integral sign is no longer valid, and the estimators in Eq. 5 and Eq. 7 generally produce incorrect results.

The fundamental idea of our method is that we can overcome this problem using a change of variables that removes the discontinuity of the integrand in θ. If such a transformation $T : \mathcal{Y} \to \mathcal{X}$ exists, then the re-parameterized integral

$$\int_{\mathcal{X}} f(x, \theta)\, \mathrm{d}x = \int_{\mathcal{Y}} f(T(y, \theta), \theta)\, |\det J_T|\, \mathrm{d}y \tag{9}$$

can be handled normally using the previously discussed estimators. Consider the example of an integral of a continuous integration kernel that is multiplied by a 1D indicator function that takes on the role of the visibility function:

$$f(x) = 1_{x>\theta}\, k(x), \quad \text{with} \int_{\mathcal{X}} k(x)\, \mathrm{d}x = 1. \tag{10}$$

Here θ is a parameter that determines the position of a discontinuity that prevents differentiation under the integral sign. We can define a new integration variable $y = x - \theta$ and use it to perform a change of variables, where $|\det J_T| = 1$:

$$I = \int_{\mathcal{X}} f(x)\, \mathrm{d}x = \int_{\mathcal{Y}} 1_{y>0}\, k(y + \theta)\, \mathrm{d}y. \tag{11}$$

Following this change, the function $1_{y>0}\, k(y + \theta)$ is differentiable with respect to θ at every point y, and we obtain

$$\frac{\partial I}{\partial \theta} \approx \frac{1}{N} \sum \frac{\partial}{\partial \theta} \frac{1_{y_i>0}\, k(y_i + \theta)}{p(y_i)}. \tag{12}$$

This change can be interpreted in the following two ways:

- Instead of integrating a function with a discontinuity whose position depends on θ, we integrate in a space where the discontinuity does not move when θ changes.
- This is equivalent to importance sampling the integral $\int f(x)\, \mathrm{d}x$ using samples $x_i(\theta) = y_i + \theta$ that *follow* the discontinuity, as shown in Figure 2.

This technique is very simple assuming that a suitable change of variables T is available: we first draw samples y_i from the density p and evaluate $f(T(y_i))$, $p(y_i)$ and the Jacobian (if any). The resulting estimate is finally differentiated using automatic differentiation.

It is imperative that our transformation does not affect the primal computation of I, which will generally rely on carefully-chosen sampling patterns. For this reason, the transformation T should be designed to yield the identity map when $\theta = \theta_0$, where θ_0 refers to the concrete parameter value for which gradients are to be evaluated. In our previous 1D example, this could be accomplished by setting

$$T(y, \theta) = y + \theta - \theta_0. \tag{13}$$

Note also that the density p from which the y_i are sampled must not depend on θ, otherwise discontinuities would be reintroduced in the integrand.

4 METHOD

We now show how to apply the technique introduced in Section 3 to practical gradient estimation in path tracers with discontinuous integrands. For the sake of clarity, we assume that all relevant integrals, including pixel integrals, have been re-parameterized over spherical domains, which incurs no loss of generality. We initially introduce the simplifying assumption that integrands have small angular support—in other words, that they vanish outside of a small neighborhood around a given direction. Then, we show how to generalize our method to arbitrary integrands.

4.1 Removing discontinuities using rotations

A typical shading integral may have arbitrarily complex discontinuities that depend on many scene parameters, and there is no simple change of variables that would be able to remove[1] them all. When integrands have a small angular support, e.g. due to pixel reconstruction filters, narrowly peaked BSDF models, or illumination from a light source with a small projected solid angle, the discontinuities typically consist of the silhouette of a single object as shown in Figure 2. This is a key assumption of our method.

Moreover, the displacement of this silhouette on S^2 for infinitesimal perturbations of θ is well-approximated using spherical rotations. Such rotations, illustrated in Figure 4a, are the spherical counterparts of translations of the planar domain (Figure 2). The smaller the support of the integrand on the sphere, the better the approximation, and for infinitely small supports the infinitesimal displacement of a point on a discontinuity can be encoded exactly using a spherical rotation. Our main simplifying assumption, then, is that there exists a suitable rotation so that the change of variables

$$I = \int_{S^2} f(\omega, \theta)\, \mathrm{d}\omega = \int_{S^2} f(R(\omega, \theta), \theta)\, \mathrm{d}\omega \tag{14}$$

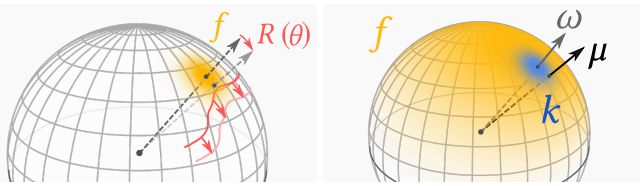
makes $f(R(\omega, \theta), \theta)$ continuous with respect to θ for each direction ω. Note that the Jacobian determinant of this change of variables is equal to 1, and that the rotation must depend on the parameter value θ for this to work.

Although more sophisticated transformations could be used, rotations are simple to compute efficiently and approximate the displacement of discontinuities well, producing gradient images that closely match ground truth as shown in Figure 3. Using a rotation R to re-parameterize the integral leads to the Monte Carlo estimate

$$E = \frac{1}{N} \sum \frac{f(R(\omega_i, \theta), \theta)}{p(\omega_i, \theta_0)} \approx I \tag{15}$$

where ω_i are sampled from the distribution p—normally the default sampling scheme that would be used in the underlying rendering algorithm (e.g. BSDF sampling). Note the use of θ_0 instead of θ to remove the dependency of the samples ω_i on θ, as discussed at the end of Section 3. Before explaining how the necessary rotations are found (Section 4.3), we turn to the more general case of integrands with large support.

[1] Note that by "remove", we refer to making the position of discontinuities independent of scene parameters rather than removing them completely.



(a) Differentiable rotation of directions

(b) Notations for our spherical convolutions

Fig. 4. **(a)** We rely on spherical rotations to re-parameterize integrands f with discontinuities such that they become differentiable. The red line represents a discontinuity in the integrand, e.g., the projected silhouette of one object of the scene. **(b)** We handle functions f with a large support (e.g. diffuse shading integrals) by introducing a spherical convolution: instead of re-parameterizing the integrand of f, we re-parameterize the integral of f times the convolution kernel k centered around a sampled direction ω.

4.2 Generalizing to functions with large support

When integrands have a large support on S^2, they generally contain a more complex arrangement of singularities that violates the simplifying assumptions made in Section 4.1, producing significant bias in gradient estimates.

Our solution to this problem relies on the property that the integral of a function f is equal to the integral of a convolution of f:

$$\int_{S^2} f(\omega)\,\mathrm{d}\omega = \int_{S^2}\int_{S^2} f(\mu)\,k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega, \tag{16}$$

where k is a spherical convolution kernel satisfying

$$\int_{S^2} k(\mu,\omega)\,\mathrm{d}\mu = 1. \quad \forall\omega \in S^2 \tag{17}$$

The convolution kernel could be any smooth distribution (e.g. a von Mises-Fisher distribution), whose concentration parameter can be set so that k has small angular support, making the inner integral of Eq. 16 compatible with the technique presented in Section 4.1. Based on these observations, we thus introduce an additional convolution to handle integrands with large support, illustrated in Figure 4b.

To numerically evaluate Eq. 16, we sample directions ω_i for the outer integral, and then sample offset directions $\mu_i \sim k(\cdot,\omega_i)$, which yields the final estimator

$$I \approx E = \frac{1}{N}\sum \frac{f(R_i(\mu_i,\theta),\theta)\;k(R_i(\mu_i,\theta),\omega_i(\theta),\theta)}{p(\omega_i(\theta),\theta)\;p_k(\mu_i)}, \tag{18}$$

whose structure is analogous to Eq. 15. The size of the spherical kernel k provides a trade-off between variance and bias of the resulting gradient estimates: for very small kernels, rotations are an excellent model for the displacement of discontinuities, but the probability of finding discontinuities within the kernel support is small, hence gradient estimates are subject to increased variance. We will revisit the choice of convolution kernels in Section 6.

4.3 Determining suitable rotations using ray tracing

Efficient determination of rotations that compensate the displacement of discontinuities with respect to scene parameters is an important building block of our differentiable renderer. Unlike previous work, we want to avoid an explicit search for silhouette edges, since this approach does not scale to large numbers of edges and high depth complexity.

A key observation of our approach is that a suitable change of variables can be found without searching for silhouette edges—in fact, it is not even necessary to know *whether* a particular integrand contains a silhouette. The only necessary information is the displacement of discontinuities—if present—with respect to infinitesimal perturbations of scene parameters.

Our previous restriction to functions with a small angular support (either directly, or via convolution) is helpful at this point: within the support of the integrand, the displacement of points on silhouette edges will closely approximate the displacement of *other positions* on the associated object. We exploit this observation by intersecting a small number of rays within the support of the integrand against the scene geometry (4 in our implementation, for details see Section 6). This number has an impact on the probability of missing a discontinuity by sampling only one of the objects of the integrand, which results in bias. Using the resulting information about their distance and surface normals, we apply a heuristic to select a single point that is likely to belong to an occluder, whose motion with respect to scene parameters matches that of the silhouette (if present). The details of this process are described in Appendix C.

We denote the projection of the selected point onto the integration domain S^2 as $\omega_P(\theta)$, and set $\omega_{P0} = \omega_P(\theta_0)$, which is the direction associated with the current parameter configuration. We then build a differentiable rotation matrix $R(\theta)$ that satisfies

$$R(\theta_0)\,\omega = \omega,\ \forall\omega \in S^2 \quad \text{and} \quad \frac{\partial}{\partial\theta}R(\theta)\,\omega_{P0} = \frac{\partial}{\partial\theta}\omega_P(\theta). \tag{19}$$

In other words: there is no rotation for $\theta = \theta_0$, and for other values it tracks the motion of the moving occluder to first order. Appendix B provides a closed-form expression for a transformation $R(\theta)$ that accomplishes this.

Importantly, our technique only relies on standard ray intersection queries, whose implementation is well studied and highly optimized on both CPUs [Wald et al. 2014] and GPUs [Parker et al. 2010], especially in the case of packets of rays with similar directions. As illustrated in Figure 5, the estimation of changes of variables is done independently for each integral at render time, and some of the rays created for this task can be reused for sampling the integrals.

5 VARIANCE REDUCTION USING CONTROL VARIATES

A naïve direct implementation of the change of variables technique described in the previous sections produces gradient estimates with significant variance. While stochastic gradient-based optimization techniques can work with noisy gradients, this leads to suboptimal convergence due to the need to take many small gradient steps. Fortunately, gradient estimators have helpful statistical properties that we can leverage using *control variates* to reduce variance substantially (Figure 6).

5.1 The statistics of gradient estimators

Our re-parameterization technique introduces variance in gradient estimates. To see why, consider the integral involving the product of f and the kernel k that arises in the inner integral of Eq. 16, where

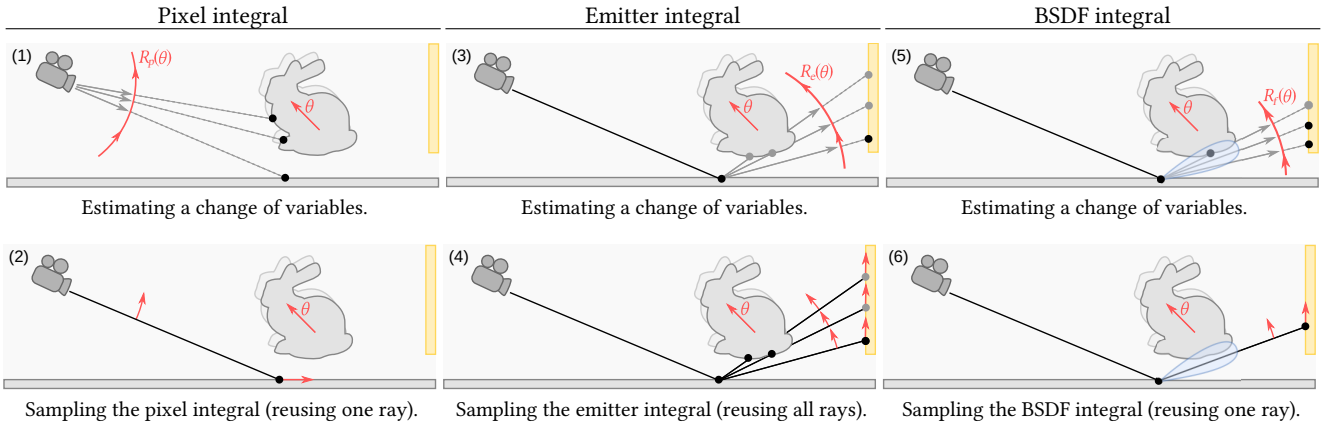


Fig. 5. Overview of our algorithm. For each integral, we intersect a small number of rays against the scene geometry and use the resulting information to construct suitable changes of variables that remove discontinuities with respect to scene parameters (1, 3, 5). These are simple local rotations (red arcs) that generally differ for each sample, and which depend on scenes parameters like the position of objects in the scene. Our method reuses some of the rays for evaluating illumination integrals (2 ,4, 6). The rotations do not affect the primal computation (i.e. ray tracing, shading) but introduce gradients that correct for the presence of discontinuities.

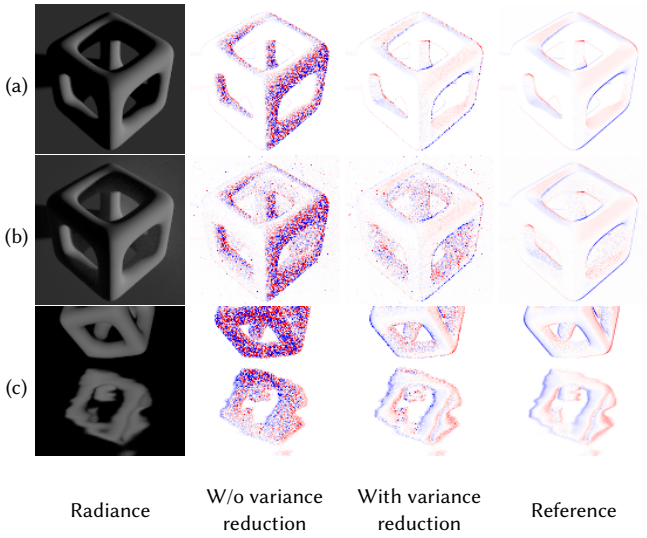


Fig. 6. The special statistical properties of our gradients estimators enable significant variance reduction without additional bias for both **(a)** direct and **(c)** indirect visibility. **(b)** shows the same scene as (a), but rendered using 3-bounce global illumination instead of direct illumination. These gradient images have been rendered with 32 pairs of correlated paths per pixel.

T is the associated change of variables:

$$E = \frac{1}{N}\sum f(T(y_i,\theta),\theta)\,\underbrace{\frac{k(T(y_i,\theta),\theta)}{k(T(y_i,\theta_0),\theta_0)}}_{=:w_i(\theta)} \tag{20}$$

Here, the weights $w_i(\theta)$ are equal to 1 in the primal computation (since $\theta_0 = \theta$), while their gradients account for the displacement of the samples $T(y_i,\theta)$. Suppose now that $f(x) = c$ is a constant function, in which case the expected value of the gradient $\mathbb{E}[\partial E/\partial\theta]$ vanishes. However, inspection of Eq. 20 reveals that individual summands in $\partial E/\partial\theta$ are generally nonzero, and that the variance of the estimator arises from the gradients of the weights w_i. However, $\mathbb{E}[\partial E/\partial\theta]$ is always equal to zero for any distribution k, and we can use this property to reduce the variance of $\partial E/\partial\theta$ using the method of control variates.

5.2 Reducing variance using control variates

The control variates method is able to reduce the variance of an estimator E if it is correlated with another estimator F, whose expected value is known. In this case, a new estimator E' can be defined as

$$E' = E + \alpha\,(F - \mathbb{E}[F]) \tag{21}$$

where $\alpha \in \mathbb{R}$ influences the variance of E' and must therefore be chosen carefully. The optimal choice of the parameter α is given by

$$\alpha = -\frac{\mathrm{Cov}(E,F)}{\mathrm{Var}(F)}. \tag{22}$$

We see that α should be negative in the case of a positive correlation of E and F and positive otherwise. In our application, this parameter is different in every integral, and accurate estimation would be prohibitive. Instead, we use approximate parameters that allow for a significant variance reduction with low performance overhead (Section 5.3). By differentiating the estimator of Eq. 20,

$$\frac{\partial E}{\partial\theta} = \frac{1}{N}\sum\frac{\partial}{\partial\theta}\Big[f(T(y_i,\theta),\theta)\,w_i(\theta)\Big], \tag{23}$$

we can see that this estimator is correlated with

$$\frac{\partial F}{\partial\theta} = \frac{1}{N}\sum\frac{\partial w_i(\theta)}{\partial\theta}, \tag{24}$$

in particular when f is a smooth function with small gradients with respect to the scene parameters θ, which we expect to be the case when rendering a 3D scene. We know that the expected value of the estimator $\frac{\partial F}{\partial\theta}$ is zero and can hence use it to define an improved

estimator for the gradient of I:

$$E' = \frac{1}{N} \sum \Big[f(T(y_i, \theta), \theta)\, w_i(\theta) + \alpha (w_i(\theta) - w_i(\theta_0)) \Big], \quad (25)$$

$$\frac{\partial E'}{\partial \theta} = \frac{1}{N} \sum \frac{\partial}{\partial \theta} \Big[f(T(y_i, \theta), \theta)\, w_i(\theta) + \alpha w_i(\theta) \Big]. \quad (26)$$

With this new estimator, we see that the variance of the gradient can be reduced completely in the case $f(x) = c$ if we set $\alpha = -c$. In practice, the variance can be reduced significantly even when f is not a constant function. The next section addresses the problem of the choice of the value α at run-time for each estimator.

5.3 Cross-reduction of the variance using pairs of paths

From Eq. 26, we observe that the optimal value of α for each integral depends on the function f and should be close to the average value of f in the integrand when f is a smooth function. As mentioned above, α must be independent of the weights $w_i(\theta)$ to avoid introducing bias in the gradient of E'.

To estimate parameters α without bias, we rely on a technique referred to as *cross-weighting scheme* in previous work that applied control variates to the computation of image-space gradients [Rousselle et al. 2016]. The key idea is to use two (or more) uncorrelated estimates, so that suitable values of α can be obtained from the other estimate without introducing bias.

Our differentiable path tracer estimates many nested integrals corresponding to each scattering event along a light path. To reduce the variance of gradient estimates, we sample *pairs* of similar paths followed by a variance cross-reduction for each pair. At each scattering event, a path accumulates radiance either via specialized direct illumination sampling strategies, or by stochastically intersecting an emitter as part of BSDF sampling. The total contribution r_i of one path can thus be written as a sum:

$$r_i = \sum_{l=0}^{\infty} f_{i,l}(\theta)\; W_{i,l}(\theta) \quad (27)$$

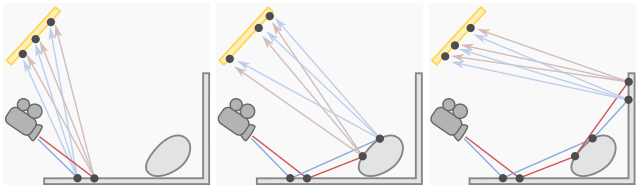
where each index l corresponds to a contribution from an emitter, $f_{i,l}(\theta)$ represents the product of emitted radiance and path throughput at vertex l, and the term $W_{i,l}(\theta)$ is the product of all associated weights $w_{i,l}(\theta)$ (Eq. 20). Because $\mathbb{E}[\partial w_i / \partial \theta] = 0$ and each weight $w_{i,l}(\theta)$ is furthermore independent from the gradient of the other weights, we have $\mathbb{E}[\partial W_{i,l} / \partial \theta] = 0$, and the final cross-reduction of variance of two paths contributions r_0 and r_1 is thus given by:

$$\begin{aligned} r' = {} & \frac{1}{2} \sum f_{0,l}(\theta)\; W_{0,l}(\theta) - f_{1,l}(\theta)\, (W_{0,l}(\theta) - W_{0,l}(\theta_0)) \\ & + \frac{1}{2} \sum f_{1,l}(\theta)\; W_{1,l}(\theta) - f_{0,l}(\theta)\, (W_{1,l}(\theta) - W_{1,l}(\theta_0)). \end{aligned} \quad (28)$$

5.4 Using partially correlated paths

The last missing piece is a specification of what constitutes a "similar" path in our method. Our goal is to sample paths that are *very correlated* so that the variance reduction is effective. A straightforward way of generating perfectly correlated paths is to simply reuse the random numbers that were used during path construction.

On the other hand, we require that there is no correlation between the estimate $f_{0,l}$ and $\partial W_{1,l}(\theta)/\partial\theta$ for all scattering orders l (and vice versa for $f_{1,l}$ and $\partial W_{0,l}(\theta)/\partial\theta$). It is possible to simultaneously satisfy both requirements by re-using the random numbers for all sampling steps *except* those that affect the weights $W_{i,l}$, i.e., the sampling of directions in the re-parameterized integrals with small angular support (Section 4.1). This yields very similar pairs of paths and is illustrated in Figure 7.



(a) Bounce 1. (b) Bounce 2. (c) Bounce 3.

Fig. 7. Our method samples correlated paths that share some of their random numbers, while others are chosen independently. The gradients associated with the resulting pairs of nearby paths (blue and red) contain uncorrelated terms that we leverage in conjunction with the technique of control variates to reduce variance substantially without adding bias.

6 IMPLEMENTATION

We implemented our system on top of Mitsuba 2 [?], in which rendering algorithms are specified in a generic form that is processed by a set of program transformations to generate concrete realizations of these algorithms. We use two transformations in particular: the first applies forward or reverse-mode automatic differentiation, and the second is a just-in-time compiler that dynamically generates kernels that efficiently evaluate the desired computation on the GPU. All experiments were conducted using a NVIDIA TITAN RTX graphics card, using OptiX [Parker et al. 2010] to perform ray-triangle intersection queries. Note that OptiX is not aware of differentiable scene parameters, hence the resulting intersection information must subsequently be converted into intersections with differentiable positions, normals, tangents and UV coordinates.

Thanks to automatic differentiation, it is simple to incorporate various differentiable scene parameters, such as vertex positions, transformation matrices for cameras, lights and objects, reflectance and displacement textures, and BSDF parameters such as albedo, roughness and index of refraction, etc.

Our method differs from a standard path tracer in that it adds extra steps, such as the estimation of changes of variables for camera, emitter and BSDF sampling, additional 3D matrix/vector products for rotating sampled directions, and the cross-reduction of the variance when accumulating radiance contributions at each bounce. All of these changes are implemented in the central path tracing loop and do not affect the rest of the system. Minor changes have been made to multi-lobed BSDFs to make then differentiable, as discussed in Appendix A. Unlike previous work [Li 2019; Li et al. 2018], we do not need to convert BSDFs into the Linearly Transformed Cosines representation [Heitz et al. 2016a].

Compared to standard path tracing, our technique adds several parameters that influence performance and the quality of gradients:

(1) The number of samples used to estimate the changes of variables in each integral. We use 4 samples (that is, 4 ray intersection

queries) for each emitter, BSDF and pixel integral. Some camera and BSDF rays are reused when sampling the underlying integrals, as shown in Figure 5, and all light samples are used to estimate the emitter integral, since they provide information about visibility that reduces the primal variance of shadows.

(2) As explained in Section 4, our gradients are more accurate in integrands with a small angular support, and we thus introduce convolutions to handle integrands that violate this assumption. In our implementation, we use convolutions when the Beckmann roughness (i.e. root mean square of slopes in the microgeometry) exceeds a value of 0.15. A similar threshold could be added for controlling the accuracy of the gradients of emitter integrals based on the solid angle of emitters at a given shading point.

(3) Our algorithm also requires a convolution kernel when spherical convolutions are used. In our implementation, we use a von Mises-Fisher distribution with a fixed concentration parameter $\kappa = 1000$. This parameter has a minor impact on the quality and variance because gradients of diffuse scattering are typically negligible compared to the gradients of direct visibility, glossy reflections and sharp shadows.

Performance. Forward rendering and backpropagation of gradients to the scene parameters usually take less than one minute for images with 512x512 pixels, using 32 pairs of paths with 3 bounces, and a few in simpler cases. The computation is roughly split across OptiX ray-triangle intersections ($\approx$ 9%), re-computing the intersections so that they are differentiable, ($\approx$ 28%), estimating discontinuities from intersections ($\approx$ 2%), rotations and von Mises-Fisher distributions ($\approx$ 9%), additional BSDF evaluation and sampling ($\approx$ 8%), as well as various additional computation for convolutions and variance reduction ($\approx$ 3%). The backpropagation step only represents approximately 8% of the total cost since the computation graph is simplified on the fly during the forward rendering pass. Altogether, our algorithm is approximately six times slower than an ordinary differentiable path tracer that does not account for discontinuities. Note that rendering gradients using edge sampling would also require differentiable intersections, BSDF evaluation, and the automatic differentiation of many quantities that depend on discontinuities. One bottleneck of our path tracer is the memory usage of the graph created for automatic differentiation on the GPU. More rays could be sampled in each kernel if this memory footprint was reduced.

We also compared the performance of our method to `redner` [Li 2019], the publicly available implementation of the edge-sampling approach by Li et al. [2018]. Although the relative performance significantly depends on the scene, it generally appears to be within the same order of magnitude. For instance, rendering and backpropagating the gradients for the scene of Figure 6c at resolution 512×512 and with 3 bounces takes 10s with our method and 12s with `redner` for 16 pairs of paths and 32 samples per pixel, respectively. Downsampling the geometry of this scene from 140k to 8k edges results in computation times of 10s (ours) and 8s (`redner`). Using an environment map increases the cost in `redner` (18s) since this makes the sampling of silhouette edges less efficient.

Importance sampling spherical convolutions. One potential challenge of spherical convolutions is that convolving an existing distribution (e.g. of a BSDF model) with a convolution kernel leads to a new distribution that has a wider support and generally cannot be expressed in closed form. In Appendix D, we derive a correction term that enables sampling of spherical convolutions using standard techniques such as BSDF importance sampling.

Since any distribution can be used for multiple importance sampling as long as this is done consistently [Guo et al. 2018; Heitz et al. 2016b], we simply fall back to the input distribution, which is in any case very similar to its convolution since we use kernels with small angular support.

Light sources. Similar to Li et al. [2018], our approach is not compatible with degenerate light sources containing Dirac delta terms (e.g. point or directional light sources), since the integrals containing visibility-related discontinuities collapse in this case.

In scenes with area lights, the associated integrands can contain two different types of discontinuities: the silhouette of the light source, and the silhouettes of the objects that partially occlude it. This is potentially a problem, since our method relies on the assumption that each integrand contains a single type of discontinuity (Section 4.1). Although the spherical convolutions described in Section 4.2 can be used to address this problem, we found that standard light sources can simply be replaced by more continuous analogues without visual changes.

In particular, we use directional lights with a smooth emission profile in the spherical domain (using vMF lobes), and area lights (e.g. rectangles or discs), whose spatially varying emission function smoothly falls off to zero approaching the boundary. When using these smooth emitters, the only remaining discontinuities in shading integrals are due to the silhouette of occluders, whose displacement can be accurately approximated using an infinitesimal rotation.

7 RESULTS AND DISCUSSION

Shape optimization. We apply our algorithm to a challenging inverse rendering problem, which entails reconstructing both geometry and materials from a set of photographs. A naïve application of gradient-based optimization tends to get stuck in local minima, and we use a multi-scale approach to avoid this problem. We begin by initializing the scene with a mesh whose shape and topology are similar to the target object, and assign a constant displacement map and texture. We then perform gradient descent using momentum, learning rate decay, maximum steps size in parameter space, and the CIELAB ΔE distance metric. We manually adjust these parameters for each optimization. Starting from the coarse initialization, we run gradient descent iterations and progressively increase the degrees of freedom by transitioning to finer resolutions of the displacement and reflectance textures every 40 steps. We have found it helpful to use reflectance textures whose resolution is below that of the displacement maps at the beginning of the optimization. In this way, shape details are reconstructed ahead of details in textures, reducing ambiguities when features in target images could be explained by an incorrect shape with a high-resolution reflectance texture. We used 300 to 500 optimization steps with 8 pairs of paths per pixel, 4 to 7 different views to reduce ambiguities (all rendered at every

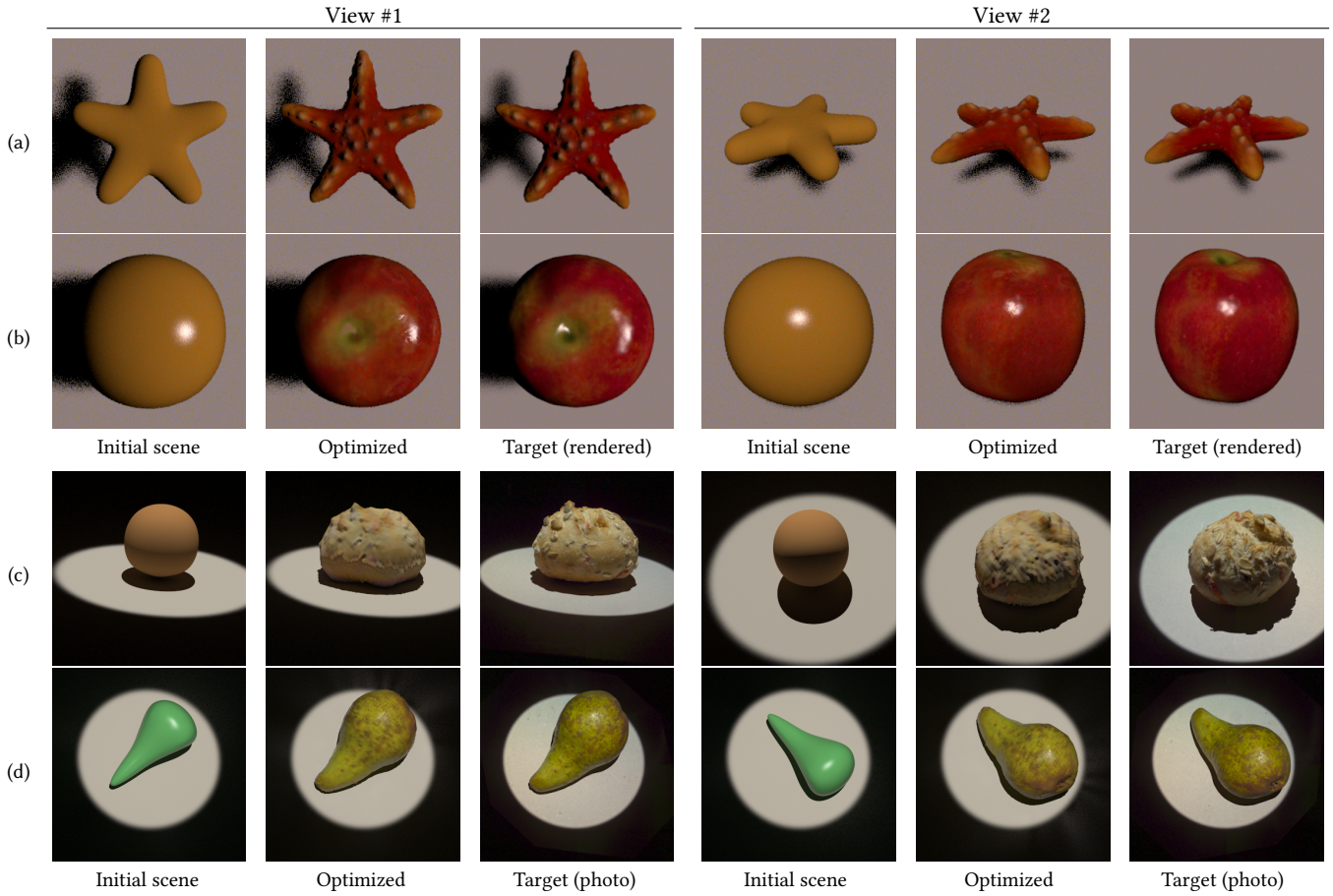


Fig. 8. We applied our technique to solve an inverse rendering problem that involves reconstructing shapes and textures from multiple reference images of an object. We used 4 to 7 views for each of these examples, of which only two views are shown in this figure. Each view is rendered at each step using 16 samples per pixel. The shapes are parameterized by displacement and texture maps, whose resolution progressively increases during gradient descent iterations. Estimating low frequency features ahead of fine-scale details is crucial to finding good solutions for this inherently ambiguous problem. (a,b): Synthetic examples with non-trivial shapes and specular reflections. (c,d): Optimization real-world photographs with calibrated camera positions. The silhouettes and shadows of the virtual objects match the reference images thanks to gradients that are introduced by these discontinuities.

step). With our implementation, each step takes approximately 1 second per view. Figure 8 shows results on two synthetic examples (starfish, apple) and real-world objects (bread, pear) captured from calibrated camera positions. The shadows of the objects provide valuable information about their shapes, as shown in Figure 9.

Many extensions of this type of reconstruction are conceivable: for instance, instead of pixel-wise losses, it would likely be preferable to use richer metrics such as structural similarity [Wang et al. 2003] or pre-trained layers from a convolutional neural network such as VGG [Simonyan and Zisserman 2015]. Our method could also be used in the context of an inverse transport network [Che et al. 2018], which is an autoencoder architecture where a rendering algorithm "decodes" parameters computed by a neural encoder network, and backpropagation then proceeds through both components.

Comparison to Li et al. [2018]. Figure 10 compares our method to `redner` [Li 2019] using low sample counts (16 pairs of paths per pixel for our renderer, and 32 paths per pixel for `redner`). Our estimators have a higher variance in cases where edge sampling is efficient and robust—in other words, for primal visibility where edges can be precomputed, and for indirect visibility in scenes with a relatively modest number of triangles. Our method produces superior estimates when rendering more complex meshes with significant geometric or depth complexity. This suggests that the strength of each method could be combined, by using edge sampling in simple scenes and for direct visibility. Both methods estimate gradients independently for each integral and could therefore coexist within the same framework.

Variance and bias of gradient estimates. Our gradients have significant variance in several cases including high-order diffuse interreflection as shown in Figure 6 (b). Our variance reduction technique requires that pairs of paths gather similar radiance contributions, and this property tends to decrease with a higher number of bounces. Increased variance for diffuse interreflection is, however, not specific to our work and was also observed by Li et al., who disable

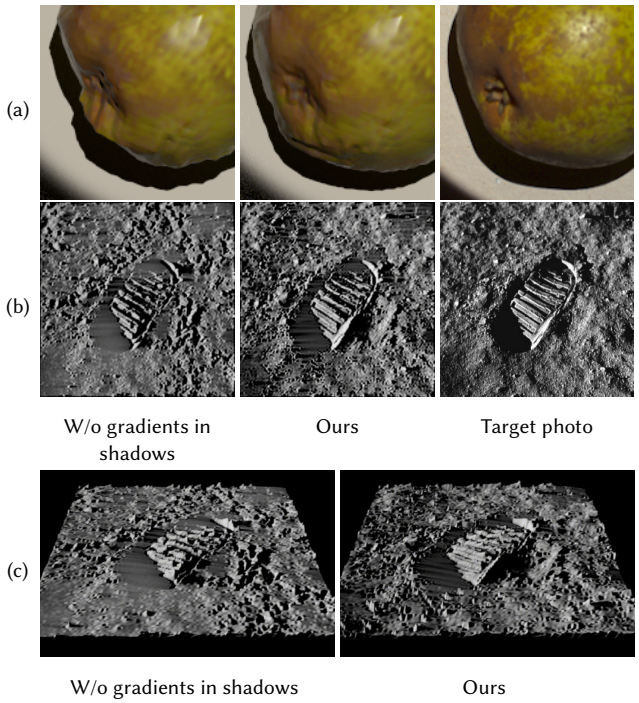


Fig. 9. Indirect visibility effects such as shadows provide important clues about the shapes of objects. Their gradients can effectively guide the optimization to accurate solutions in inverse rendering applications. **(a)**: Detail of our multi-view reconstruction (Figure 8). **(b)**: Reconstruction of a diffuse surface from a single photo of the surface of the Moon modeled as a height field. The orientation of the light is fixed, and its intensity is estimated from the mean luminance in the rendered and target images. **(c)**: The same reconstruction viewed from an oblique angle.

the associated gradients in `redner`. Other cases exhibiting variance include rapidly changing radiance values, e.g. in very sharp shadows in locations where an occluder directly touches another surface that is being shadowed. In general, rendering of gradients introduces a number of additional complications compared to a primal rendering, and high-quality gradient estimation in scenes with complex light transport remains a challenging problem.

To overcome certain limitations of previous work, our technique relies on simplifying assumptions, which introduce a controllable amount of bias in gradients estimates. For a fixed convolution kernel size, our estimators are also not consistent: they do not converge to the correct gradients when more rays are traced. Examples of scenes with visible bias can be found in Figure 11. While it is generally assumed that stochastic gradient descent requires unbiased gradients, accurate and low-variance gradients are preferable to unbiased but uninformative gradients for approaching a local minimum within few steps. However, bias could potentially deteriorate the convergence when approaching a solution during the last steps of an optimization. The optimal trade-off between variance and bias remains unclear and deserves further study.

Other limitations. Our path tracer relies on several simplifying assumptions that may not hold in particular scenes—for instance,

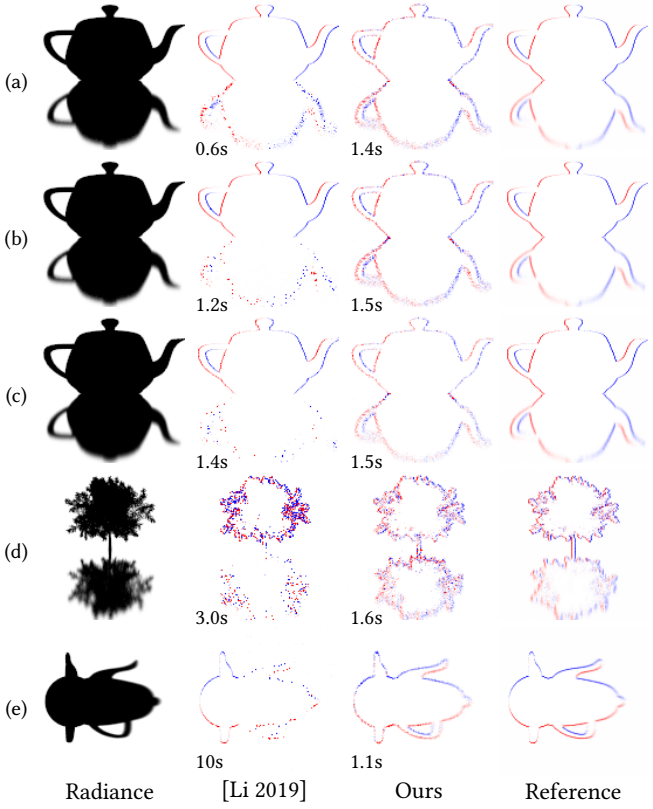


Fig. 10. Comparison between our method and `redner`, the publicly available implementation by Li et al. [Li et al. 2018]. Gradients are computed with respect to translations of the shown objects. (a) A diffuse black teapot (16K edges) on a rough metallic surface. (b) The same teapot with 3 levels of subdivision (1M edges). (c) The subdivided teapot with small amount of vertex displacement to simulate geometric roughness. (d) A tree model with quadrilateral leaves (5M edges). (e) Shadow of the subdivided teapot on a diffuse plane, seen from the top. The performance of edge sampling degrades with the number of edges for indirect visibility (i.e. specular reflection and shadows). Our method remains independent of both geometric and depth complexity, producing high-quality gradients even in challenging scenes with many silhouette edges. The specified times include the forward rendering and backpropagation passes but exclude scene loading time.

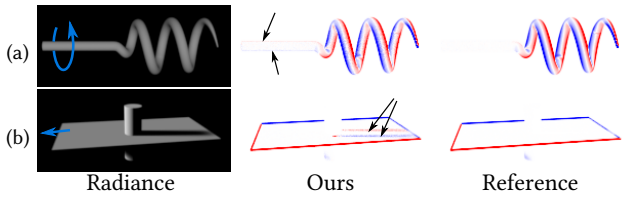


Fig. 11. Our method relies on approximations that introduce bias when a reparameterization does not fully remove the parameter-dependent displacement of a discontinuity. This becomes apparent in pixels whose correct gradient is zero (black arrows). The magnitude of the bias is generally small (Fig. 3) but it is not bounded and can be high in some pixels and configurations (*e.g.*, up to 35% of the maximum gradient in example (b)).

the assumption that there is a single type of discontinuity within the support of an integrand may not hold when two discontinuities are very close to each other. To our knowledge, explicit edge sampling is the only feasible solution in such difficult cases, although they did not manifest in the kinds of optimization problems conducted as part of this work. As in previous work, our renderer does not support perfectly specular materials and degenerate light sources containing Dirac delta functions. Generalizing our method to such cases would be an interesting topic for future work.

8 CONCLUSION

We introduced a novel technique for differentiating pixel values with respect to arbitrary scene parameters based on a re-parameterization approach that removes discontinuities in the integrands, enabling differentiation of standard Monte Carlo estimates. Our differentiable path tracer requires three building blocks that can be implemented in existing systems without major modifications of the underlying architecture: the estimation of differentiable rotations using several ray intersection queries, a sampling technique for spherical convolutions, and a variance reduction technique that traces correlated pairs of light paths.

Our path tracer improves on previous work for estimating gradients in shadows and glossy reflections. In particular, it bypasses the key limitation of prior work that requires position samples on visible silhouette edges, allowing our method to scale to scenes with high geometric and depth complexity, such as high-resolution displacements and trees with an intricate arrangement of silhouette edges.

We used our path tracer to optimize meshes using differentiable displacement maps, demonstrating that our technique enables detailed reconstruction of shapes from photos while jointly optimizing suitable BRDF parameters. Future work on improving the performance and robustness of differentiable renderers would be beneficial to many fields of research including appearance capture, computational material design, and computer vision.

ACKNOWLEDGMENTS

We would like to thank Merlin Nimier-David, Tizian Zeltner and Delio Vicini for their support and feedback throughout this project. We also thank altphotos.com for the wood texture. This research was supported by the Inria-EPFL International Lab.

REFERENCES

Marilyne Andersen, Siân Kleindienst, Lu Yi, Jaime Lee, Magali Bodart, and Barbara Cutler. 2008. An intuitive daylighting performance analysis and optimization approach. *Building Research & Information* 36, 6 (2008), 593–607.

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

Serge Belongie. 2019. Rodrigues' Rotation Formula. From MathWorld — A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/RodriguesRotationFormula.html

Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2018. Inverse Transport Networks. (2018). http://arxiv.org/abs/1809.10820

M. de La Gorce, N. Paragios, and D. J. Fleet. 2008. Model-based hand tracking with texture, shading and self-occlusions. In *2008 IEEE Conference on Computer Vision and Pattern Recognition.* 1–8.

Amaël Delaunoy and Emmanuel Prados. 2011. Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems Dealing with Visibility. *International Journal of Computer Vision* 95, 2 (01 Nov. 2011), 100–123.

P. Gargallo, E. Prados, and P. Sturm. 2007. Minimizing the Reprojection Error in Surface Reconstruction from Images. In *ICCV 2007.* 1–8.

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An Evaluation of Computational Imaging Techniques for Heterogeneous Inverse Scattering. In *Computer Vision – ECCV 2016.* Springer International Publishing, Cham, 685–701.

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

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

Yu Guo, Miloš Hašan, and Shaung Zhao. 2018. Position-Free Monte Carlo Simulation for Arbitrary Layered BSDFs. *ACM Trans. Graph.* 37, 6 (2018).

Milovš Hašan and Ravi Ramamoorthi. 2013. Interactive Albedo Editing in Path-traced Volumetric Materials. *ACM Trans. Graph.* 32, 2 (April 2013).

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016a. Real-time Polygonal-light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.* 35, 4 (July 2016).

Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. 2016b. Multiple-scattering microfacet BSDFs with the Smith model. *ACM Transactions on Graphics* 35 (July 2016).

André Jalobeanu, Frank O. Kuehnel, and John C. Stutz. 2004. Modeling Images of Natural 3D Surfaces: Overview and Potential Applications. *2004 Conference on Computer Vision and Pattern Recognition Workshop* (2004).

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150.

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

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 531–540.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Trans. Graph.* 34, 4 (2015).

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.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. *ACM Trans. Graph.* 32, 4 (2013).

Tzu-Mao Li. 2019. Redner. https://github.com/BachiLi/redner.

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.

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018. Paparazzi: Surface Editing by way of Multi-View Image Processing. *ACM Transactions on Graphics* (2018).

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. (2019). arXiv:1904.01786 http://arxiv.org/abs/1904.01786

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *Computer Vision – ECCV 2014.* Springer International Publishing, Cham.

Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. 2018. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. (2018). arXiv:1806.06575 http://arxiv.org/abs/1806.06575

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38, 6 (Nov. 2019).

Matt Olson and Hao Zhang. 2006. Silhouette Extraction in Hough Space. *Comput. Graph. Forum* 25 (Sept. 2006).

Marios Papas, Christian Regg, Wojciech Jarosz, Bernd Bickel, Philip Jackson, Wojciech Matusik, Steve Marschner, and Markus Gross. 2013. Fabricating Translucent Materials Using Continuous Pigment Mixtures. *ACM Trans. Graph.* 32, 4, Article 146 (July 2013).

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 *(SIGGRAPH '10).*

Gustavo Patow and Xavier Pueyo. 2003. A Survey of Inverse Rendering Problems. *Computer Graphics Forum* 22, 4 (2003), 663–687.

Felix Petersen, Amit Bermano, Oliver Deussen, and Daniel Cohen-Or. 2019. Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer. (2019). arXiv:1903.11149 http://arxiv.org/abs/1903.11149

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A First-order Analysis of Lighting, Shading, and Shadows. *ACM Trans. Graph.* 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 *(ICCV '15).* IEEE Computer Society.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space Control Variates for Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (Dec. 2016).
D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA, USA, Chapter Learning Internal Representations by Error Propagation, 318–362.
Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. 2014. High-contrast Computational Caustic Design. *ACM Trans. Graph.* 33, 4 (July 2014).
K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations.*
V. N. Smelyansky, R. D. Morris, F. O. Kuehnel, D. A. Maluf, and P. Cheeseman. 2002. Dramatic Improvements to Feature Based Stereo. In *Computer Vision — ECCV 2002.* Springer.
Chia-Yin Tsai, Aswin C. Sankaranarayanan, and Ioannis Gkioulekas. 2019. Beyond Volumetric Albedo – A Surface Optimization Framework for Non-Line-Of-Sight Imaging. In *CVPR 2019.*
Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.
Zdravko Velinov, Marios Papas, Derek Bradley, Paulo Gotardo, Parsa Mirdehghan, Steve Marschner, Jan Novák, and Thabo Beeler. 2018. Appearance Capture and Modeling of Human Teeth. *ACM Trans. Graph.* 37, 6 (Dec. 2018).
Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Trans. Graph.* 33, 4 (July 2014).
Z Wang, Eero Simoncelli, and Alan Bovik. 2003. Multiscale structural similarity for image quality assessment. *Conference Record of the Asilomar Conference on Signals, Systems and Computers* 2.
R. E. Wengert. 1964. A Simple Automatic Derivative Evaluation Program. *Commun. ACM* 7, 8 (Aug. 1964), 463–464.
E Woodcock, T Murphy, P Hemmings, and S Longworth. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, Vol. 557.
Shaung Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling Scattering Parameters for Rendering Anisotropic Media. *ACM Trans. Graph.* 35, 6 (2016).

A CORRECTING NON-DIFFERENTIABLE SAMPLING PROCEDURES

Differentiable mappings from random numbers to samples from a given probability density function are widely used in the context of rendering. Common examples are sampling techniques for microfacet BSDFs or points on polygonal shapes (e.g. area lights). However, some sampling methods rely on non-differentiable operations as in the case of sampling multi-lobed BSDF models (e.g. reflection and refraction of a dielectric), sampling a triangle in a mesh proportional to surface area, or sampling an interaction within a heterogeneous participating medium using a technique like Woodcock tracking [Woodcock et al. 1965]. The value of a sample is typically computed from a comparison between a random number and a threshold value that depends on some scene parameters θ, and in standard implementations the value of the resulting sample is not differentiable with respect to θ. This problem can be solved will minor changes of the code, by an explicit multiplication by a weight that depends on the probability of the parameter θ, as shown in Figure 12.

```
def Sample(θ):
    r = Rand();
    if r < f(θ) then
        x = SampleA();
        p = pdfA(s);
    else
        x = SampleB();
        p = pdfB(s);
    return x, p;
```

(a) Non-differentiable sampling

```
def Sample(θ, θ0):
    r = Rand();
    if r < f(θ0) then
        x = SampleA();
        p = pdfA(s);
    else
        x = SampleB();
        p = pdfB(s);
    p *= f(θ) / f(θ0);
    return x, p;
```

(b) Differentiable sampling

Fig. 12. (a): Importance sampling a function $f(\theta)\, f_A(x) + (1 - f(\theta))\, f_B(x)$ can result in non-differentiable importance sampling weighs p with respect to θ. (b): Such sampling procedures can be easily made differentiable by importance sampling using a fixed parameter θ_0 (that is equal to θ at run time) and by multiplying by a sampling weight involving $f(\theta)$.

B DIFFERENTIABLE ROTATION MATRICES

Our differentiable path tracer requires differentiable rotation matrices R built from two directions ω_a and ω_b, such that $R\omega_a = \omega_b$. Moreover, we need to build such matrices in the special case $\omega_a = \omega_b$, that is, when these matrices are equal to the identity matrix but their gradients depend on both ω_a and ω_b. The widely used Rodrigues' rotation formula [**?**] is not defined in this case because it involves a normalized rotation axis. The norm of the rotation axis can be factored out from this formula and simplified, leading to a more general expression:

$$R = \alpha I + [\beta]_x + \frac{1}{1+\alpha}\beta^T\beta. \tag{29}$$

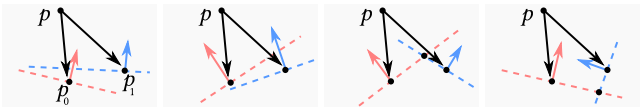
with $\alpha = \omega_a \cdot \omega_b$, $\beta = \omega_a \times \omega_b$ and

$$[u]_x = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{30}$$

The resulting matrices are differentiable in the case $\omega_a = \omega_b$.

C SELECTING OCCLUDERS

Our technique relies on the assumption that the discontinuities in a given integrand consist of the silhouette of a single object that masks either another object, another part of the same object or the background (Section 4.1). The integrand is sampled using four samples, which yields one or more surface intersections. Our reparameterization technique then requires selecting a point that likely belongs to the occluding object among these intersections. In our implementation, we address this problem by first choosing two points that belong to different objects (when applicable) based on object IDs, so that a discontinuity necessarily exists between these points. Then, our technique constructs two planes tangent to the surfaces at these points, and chooses the occluder based on the masking between these planes and the intersections. The different cases are shown in Figure 13. Our results suggest that this first-order estimation of occlusion between objects is sufficiently robust for gradient estimations in most practical cases. It could be improved in various ways, for instance by considering higher-order approximations of the surfaces and by combining information from all the intersections. One benefit of this framework is that it is able to compute gradients at the intersection of two interpenetrating objects that create an *implicit* edge (i.e. one that is not explicitly represented in the geometry) in contrast to previous work.



(a) Simple occlusion (b) Simple occlusion (c) Intersection between objects (d) Intersection between objects

Fig. 13. From a pair of surface points p_0 and p_1 that are visible from a point p, we estimate the occlusion between the corresponding objects using first-order surface approximations from the normals at p_0 and p_1. Figures (a) and (b) show cases where one plane occludes the other intersection point from p. Figures (c) and (d) illustrate the case of an intersection between objects that can be estimated from the intersection of the planes.

D SAMPLING CONVOLUTIONS

We rely on spherical convolutions to compute accurate gradients in integrands with large support. The outer integral of Eq. 16, that is, the integral of $f \circledast k$, must be importance sampled using a probability density function (the function p in Eq. 18), which should ideally be proportional to $f \circledast k$. When f is the product of a BSDF and the incident illumination, a natural choice would e.g. be to importance sample the convolution of the BSDF with the kernel k. Unfortunately, such convolutions generally do not have a closed-form expression. Fortunately, the default sampling strategy that renderers implement for f is a good candidate, as $f \circledast k$ and f are very similar functions when k is a narrowly peaked convolution kernel.

One remaining problem is that the support of the sampling distribution must cover the support of the integrand $f \circledast k$. In the case of BRDF sampling, the standard sampling procedure for f only samples directions on the hemisphere, but the support of $f \circledast k$ is wider than the hemisphere, resulting in a small loss of energy when sampling a convolution:

$$\int_{S^2} f(\omega)\,\mathrm{d}\omega = \int_{\mathcal{H}} f(\omega)\,\mathrm{d}\omega \tag{31}$$

$$= \int_{S^2}\int_{S^2} f(\mu)\,k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega \tag{32}$$

$$> \int_{\mathcal{H}}\int_{S^2} f(\mu)\,k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega, \tag{33}$$

where $\mathcal{H}$ is the upper hemisphere and S^2 denotes the unit sphere. We solve this problem by introducing a correction term that compensates the lack of sampling of the lower hemisphere and allows for unbiased convolution estimates using the standard sampling procedures on hemispherical integral. This correction term is given by

$$C(\mu) = \int_{\mathcal{H}} k(\mu,\omega)\,\mathrm{d}\omega, \tag{34}$$

and by replacing f by f/C in the inner integral we obtain

$$\int_{\mathcal{H}}\int_{S^2} \frac{f(\mu)}{C(\mu)}\,k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega = \int_{S^2}\int_{\mathcal{H}} \frac{f(\mu)}{C(\mu)}\,k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega \tag{35}$$

$$= \int_{S^2} \frac{f(\mu)}{C(\mu)} \int_{\mathcal{H}} k(\mu,\omega)\,\mathrm{d}\mu\,\mathrm{d}\omega \tag{36}$$

$$= \int_{S^2} f(\mu)\,\mathrm{d}\mu. \tag{37}$$

The correction term $C(\mu)$ is a well-behaved function that only depends on the convolution kernel, with $C(\mu) = \frac{1}{2}$ at grazing angles and $C(\mu) \approx 1$ for directions inside the hemisphere. We precompute this correction factor for the convolution kernel and sample Eq. 35 using the default strategy that the renderer used to handle the integrand in f.