Unbiased Warped-Area Sampling for Differentiable Rendering

SAI PRAVEEN BANGARU, Massachusetts Institute of Technology
TZU-MAO LI, Massachusetts Institute of Technology
FRÉDO DURAND, Massachusetts Institute of Technology

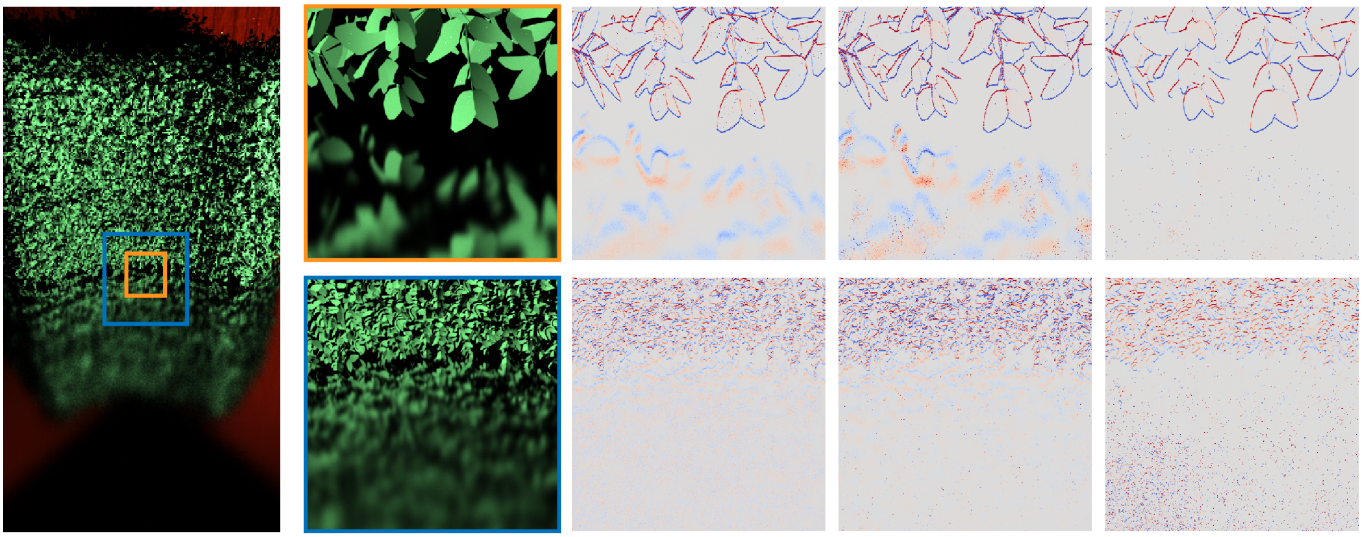


Fig. 1. Differentiable rendering computes derivatives of the light transport equation. To differentiate with the existence of visibility, recent physically-based differentiable renderers require either explicitly finding boundary points [Li et al. 2018; Zhang et al. 2020], or approximating the boundary contribution through heuristics [Loubet et al. 2019]. We develop from first principles an unbiased estimator that computes the boundary contribution from interior (area) samples. Our approach can be easily integrated with existing importance sampling methods and computes accurate and low variance gradients. For instance, the *edge sampling method* [Li et al. 2018] finds it difficult to consistently sample boundary points that contribute to the derivative in the soft reflection, especially because of the high complexity of the scene. Our method, on the other hand, uses samples from a standard path tracer and takes advantage of BSDF and light source importance sampling to compute a robust estimate for the derivative. We validate our derivatives against the finite difference image computed w.r.t the hedge's translation in the upward direction. Both our method and edge sampling used an equal number of samples.

Differentiable rendering computes derivatives of the light transport equation with respect to arbitrary 3D scene parameters, and enables various applications in inverse rendering and machine learning. We present an unbiased and efficient differentiable rendering algorithm that does not require explicit boundary sampling. We apply the divergence theorem to the derivative of the rendering integral to convert the boundary integral into an area integral. We rewrite the converted area integral to a form that is suitable for Monte Carlo rendering. We then develop an efficient Monte Carlo sampling algorithm for solving the area integral. Our method can be easily plugged into a traditional path tracer and does not require dedicated data structures for sampling boundaries.

We analyze the convergence properties through bias-variance metrics, and demonstrate our estimator's advantages over existing methods for some synthetic inverse rendering examples.

CCS Concepts: • **Computing methodologies** → *Computer vision*; **Rendering**; **Visibility**.

Additional Key Words and Phrases: inverse graphics, differentiable rendering, light transport, differentiating visibility

ACM Reference Format:
Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 245 (December 2020), 18 pages. https://doi.org/10.1145/3414685.3417833

Authors' addresses: Sai Praveen Bangaru, Massachusetts Institute of Technology, Cambridge, MA, sbangaru@mit.edu; Tzu-Mao Li, Massachusetts Institute of Technology, Cambridge, MA, tzumao@mit.edu; Frédo Durand, Massachusetts Institute of Technology, Cambridge, MA, fredo@mit.edu.

0730-0301/2020/12-ART245
https://doi.org/10.1145/3414685.3417833

1 INTRODUCTION

Differentiable rendering – the task of computing derivatives of the light transport equation [Kajiya 1986] with respect to scene parameters such as camera position, triangle mesh positions, and texture parameters, has become increasingly important for solving inverse rendering problems and training 3D deep learning models. The discontinuities introduced by visibility pose a central challenge

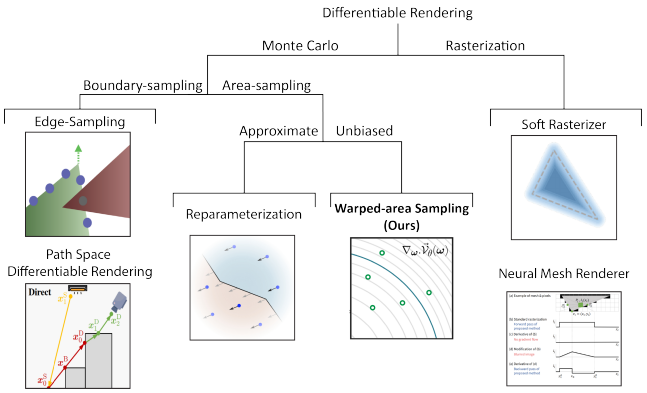


Fig. 2. Taxonomy of differentiable rendering. Both boundary sampling techniques rely on complex importance sampling data structures. Li et al. [2018] use a 6D Hough tree to find silhouettes and Zhang et al. [2020] pre-compute a spatio-angular photon map in order to find important segments. In contrast, the reparameterization method (Loubet et al. [2019]) is lightweight, and only needs to compute a rotation on-the-fly during the standard Monte Carlo rendering process, but it is biased. Our technique retains the simplicity and flexibility of the reparameterization method, while solving its bias problem.

to differentiable rendering because their derivatives have measure zero and cannot be sampled by traditional Monte Carlo methods.

Various approaches have been proposed to resolve the visibility challenges. We roughly group them into three categories (Fig. 2):

- **Rasterization.** Many deterministic differentiable rasterizers approximate visibility and ignore higher-order transport such as shadow and global illumination (e.g., [Kato et al. 2018; Loper and Black 2014]).
- **Boundary sampling.** These methods [Li et al. 2018; Zhang et al. 2020, 2019] explicitly integrate over the discontinuities that occur at object silhouettes. They can compute unbiased gradients of pixel colors with respect to the scene parameters, while taking higher-order transport into consideration.
- **Area sampling.** Loubet et al. [2019] introduced a fast approximation to boundary sampling by sampling the *area* instead of the silhouette, through tracing *auxiliary rays* inside a unidirectional path tracer, and detecting silhouettes using heuristics. They then reparameterize the integral to eliminate discontinuities by rotating the integration domain.

The boundary sampling approaches, while producing unbiased results, are usually more involved in the implementation and are less efficient, since they cannot rely on the traditional solid angle sampling infrastructures of traditional renderers. For primary visibility, the silhouette of 3D objects can be precomputed given a camera position. However, for secondary visibility, sampling from the object silhouettes given a shading point is computationally challenging and requires expensive data structure queries for finding the silhouette [Hertzmann and Zorin 2000; Li et al. 2018; Olson and Zhang 2006; Sander et al. 2000]. Furthermore, boundary sampling approaches first sample points on the edge, then connect them to other light path vertices. This connection makes edges inside mirror reflections particularly challenging to sample.

On the other hand, all the current alternatives that are not based on edge sampling are biased and do not converge to the target solution.

We propose a simple and practical solution for differentiable rendering using **area sampling**. By deriving the area integral from first principles, we show an efficient, consistent estimator for the derivative as well as a version that produces an unbiased estimate. Like Loubet et al.'s approach, our method does not require dedicated data structures for selecting silhouette edges and can be easily integrated into a traditional unidirectional path tracer by tracing auxiliary rays. Unlike Loubet et al.'s approach, our method does not introduce approximation to the integral and produces consistent or unbiased estimates.

Our key insight is that we can convert the integral over the object silhouette to an area integral by applying the divergence theorem. We show that, as long as the resulting integrand of the area integral is continuous and matches the values of the silhouette integral at the boundaries, the two integrals are equivalent. We also show that, under this interpretation, Loubet et al.'s method introduces bias, since the integrands at the silhouette do not match for the two integrals.

Given the continuity and boundary conditions, there are infinitely many functions that satisfy the constraints. However, since we want to avoid boundary sampling in the first place, we are solving a difficult boundary interpolation problem without prior knowledge of the boundaries, while having to match the contribution at the boundaries. To resolve this, we construct a smooth area integral by convolving a boundary-matching but discontinuous field. We design the convolution such that it converges to the values of the contribution at the boundaries. We then develop a sampling algorithm to produce consistent estimates for this convolution, followed by an unbiased version that uses Russian roulette de-biasing[1]. We also employ antithetic sampling and control variates to reduce the variance introduced by the convolution.

Experiments show that our sampling algorithm introduces comparable variance to Loubet et al.'s method but has almost zero bias with our consistent estimator (which reduces to zero with our unbiased version). This leads to a more accurate estimation in inverse rendering problems without the extra computational overhead of edge-sampling methods.

Our contributions are:

- We connect the silhouette integral that occurs in differentiable rendering, and the area integral in the traditional rendering equation, by applying the divergence theorem.
- We identify a family of area integrals useful for handling the discontinuities in differentiable rendering through convolution.
- We derive an efficient and practical algorithm to sample these area integrals. Our method can be easily plugged into a unidirectional path tracer.

[1]In practice, this leads to potentially unbounded memory usage, and we impose an upper bound truncation, like any path tracer employing Russian roulette (Section 5.5).

2 RELATED WORK

Light path derivatives. In light transport simulation, derivatives with respect to light path parameters are often used by rendering algorithms to guide sampling and reconstruction [Arvo 1994; Chen and Arvo 2000; Mitchell and Hanrahan 1992; Ramamoorthi et al. 2007; Shinya et al. 1987; Ward and Heckbert 1992; Wu et al. 2020]. In more recent work, derivatives are used in Markov chain Monte Carlo algorithms for guiding the mutation [Hanika et al. 2015; Jakob and Marschner 2012; Kaplanyan et al. 2014; Li et al. 2015; Luan et al. 2020; Rioux-Lavoie et al. 2020]. In contrast, we are interested in computing derivatives of light transport contributions with respect to arbitrary parameters, including scene parameters such as camera pose and triangle vertex positions.

Differentiable Rendering in Graphics and Vision. There is a strong demand in both graphics and vision communities for a general-purpose differentiable renderer. Early graphics and vision algorithms often used specialized differentiable renderers for their specific inverse graphics problems (e.g., [Blanz and Vetter 1999; Jalobeanu et al. 2004; Patow and Pueyo 2003; Smelyansky et al. 2002]). There is also an increasing interest in including a differentiable rendering *layer* inside a machine learning framework (e.g., [Aittala et al. 2016; Genova et al. 2018; Li et al. 2019; Liu et al. 2017]).

The first general-purpose differentiable renderers focused on approximating the primary visibility and supported a limited set of material models [de La Gorce et al. 2011; Kato et al. 2018; Liu et al. 2019; Loper and Black 2014; Rhodin et al. 2015].

Differentiable Light Transport Algorithms. Recently, Li et al. [2018] proposed the first unbiased Monte Carlo solution for computing the gradients of pixel color with respect to scene parameters. In addition to correctly computing the gradients with respect to local lighting and camera projection, their method also works for high-order transport phenomena such as shadows and global illumination. They observed that, after pixel prefiltering, the average pixel color changes continuously with respect to geometry parameters, making the rendering operation differentiable. Zhang et al. [2020; 2019] further generalized the idea to handle participating media and path-space rendering. Nimier-David et al. [2020] derived a new differential light transport formulation that reduces the memory requirements by propagating the derivative quantities from the camera to light sources. Our method can be used together with Nimier-David et al.'s formulation to handle the visibility gradient.

Li et al.'s method finds the *edges* that form the silhouette of the 3D scene in order to explicitly sample the Dirac delta signals that appear when differentiating the discontinuities inside an integral. However, this introduces significant overhead in the sampling infrastructure. To improve the efficiency, Loubet et al. [2019] proposed an approximated algorithm that does not require sampling points on the edges, by reparameterizing the integral through a rotation matrix constructed by heuristics. Their method achieves lower variance than Li et al.'s edge sampling, at the cost of increased bias, which can have pronounced effects on stochastic gradient optimization.

Our method combines the benefits of both edge sampling and area sampling approaches to achieve unbiasedness and efficiency. More importantly, we show that area sampling alone can achieve unbiased results. This allows us to sample the silhouette integral using a regular unidirectional path tracer, without the expensive edge selection procedure, while maintaining correctness.

Table 1. Notation

I	$\triangleq$	Rendering integral over domain Ω
Ω	$\triangleq$	Domain of 3 dimensional unit vectors $\boldsymbol{\omega}$
ω, ω'	$\triangleq$	Unit vector representing a direction in 3D space
$\mathbf{x}, \mathbf{y} \cdots$	$\triangleq$	3D scene points
$\partial_\theta f$	$\triangleq$	Partial derivative of scalar f w.r.t vector or scalar θ
$\partial_{\mathbf{v}} \mathbf{u}$	$\triangleq$	Jacobian of the mapping $\mathbf{v} \rightarrow \mathbf{u}$
$\nabla_{\mathbf{x}} . \mathbf{f}$	$\triangleq$	Divergence of the field $\mathbf{f}$

Sensitivity analysis. In the particle transport literature, Roger et al. [2005] derived a differential Monte Carlo estimator for integrals with deformable domains. They also realized the relationship between the boundary integral and the area integral. However, they interpolated boundary velocities by minimizing the Dirichlet energy. This strategy is not applicable in our case, since the minimization requires explicit enumeration of the boundaries, which is what we set out to avoid in the first place.

3 AREA FORMULATION OF DIFFERENTIABLE RENDERING

Our goal is to develop a consistent or unbiased area-based sampling method for computing the derivatives of the rendering equation. We achieve this by applying an identity of vector calculus (the *divergence theorem*) to reformulate the boundary integral as an integral over the interior supplemented by a vector field we refer to as the warp field. We show that the warp field needs to be continuous and be consistent with the derivative of the boundary points. We then pick a specific family of warp fields that depends only on quantities readily available in the context of the rendering integral.

In Section 4, we show that we can augment a unidirectional path tracer to sample the converted area integral by tracing auxiliary rays at each bounce to construct an estimate for the warp field.

3.1 Boundary Integral in Differentiable Rendering

Our goal is to compute the gradient of the rendering integral:

$$I = \int_D f(\omega; \theta) \mathrm{d}\omega, \tag{1}$$

for some domain D (e.g., the hemispherical domain), where the function f depends on some scene parameters θ such as the triangle vertex positions or material parameters. For differentiable rendering, we are interested in the derivative $\partial_\theta I$.

As shown in previous work [Li et al. 2018], direct Monte Carlo sampling using $\partial_\theta f(\mathbf{x}, \theta)$ is not unbiased since f is discontinuous with respect to θ, and the derivatives include Dirac delta signals. Instead, we need to rewrite the integral to eliminate the Dirac deltas for applying Monte Carlo integration with area-based sampling.

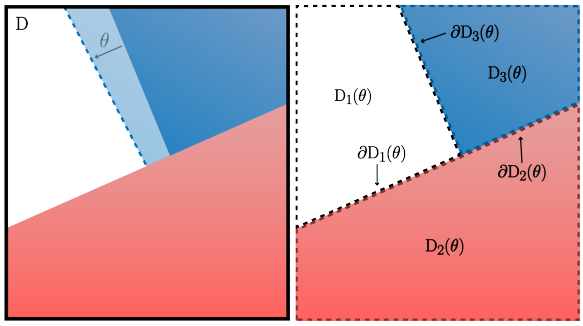


Fig. 3. Differentiating boundary movements. Our goal is to compute the derivative of the average color inside domain D with respect to some geometric parameter θ. **(a)** shows an example of the geometric contents of a pixel, **(b)** illustrates how we partition the domain D into disjoint regions, such that all the discontinuities are at the boundaries $\partial \mathrm{D}_i(\theta)$. We can then properly take the change of the boundaries into consideration when computing derivatives of discontinuous functions inside the integrals.

To formalize this process, we take inspiration from previous differentiable rendering work [Li 2019; Zhang et al. 2019] by partitioning the domain D into disjoint regions $D_0(\theta), D_1(\theta), D_2(\theta), \cdots$, such that $f(x;\theta)$ is continuous within the boundaries of each piece, and all the discontinuities are at the boundaries of the domain (Fig. 3). Therefore the domains D_i are dependent on the scene parameters θ:

$$\frac{\partial \mathcal{I}}{\partial \theta} = \sum_i \frac{\partial}{\partial \theta} \int_{\mathrm{D_i}(\theta)} f(\omega;\theta)\mathrm{d}\omega. \tag{2}$$

Importantly, the partition is only done for our derivation, and we do not explicitly clip the geometry to form the partition.

Now that the integrands in the integrals are continuous, we can rewrite the differential integrals by measuring the change of the boundary of the domain $D_i(\theta)$ with respect to parameter θ. In 1D this is the Leibniz's integral rule, which can be derived from the fundamental theorem of calculus ($\frac{\partial}{\partial \theta}\int_0^{b(\theta)} dx = \partial_\theta b(\theta)$). In higher-dimensional spaces, this is characterized by the Reynolds transport theorem [1903] (see Flanders [1973]'s article for a shorter proof), widely used in integrals that arise in fluid dynamics where the fluid flow expands or contracts. The definition of Reynolds transport theorem directly yields:

$$\frac{\partial \mathcal{I}}{\partial \theta} = \sum_i \int_{\mathrm{D_i'}(\theta)} \frac{\partial f(\omega;\theta)}{\partial \theta}\mathrm{d}\mathrm{D}_i'(\theta) + \sum_i \oint_{\partial \mathrm{D_i}(\theta)} f(\omega;\theta)(\partial_\theta \omega \cdot \hat{\mathbf{n}})\mathrm{d}\partial \mathrm{D}_i(\theta) \tag{3}$$

where the second term describes the rate at which the domain expands or contracts over elements of the boundary $\partial \mathrm{D}_i$, and the first term accounts for the continuous part of f, and $\mathrm{D}_i' = \mathrm{D}_i - \partial \mathrm{D}_i$. $\hat{\mathbf{n}}$ is the outward pointing normal vector at the boundary. We call the first term the **interior derivative integral** and the second term the **boundary derivative integral**.

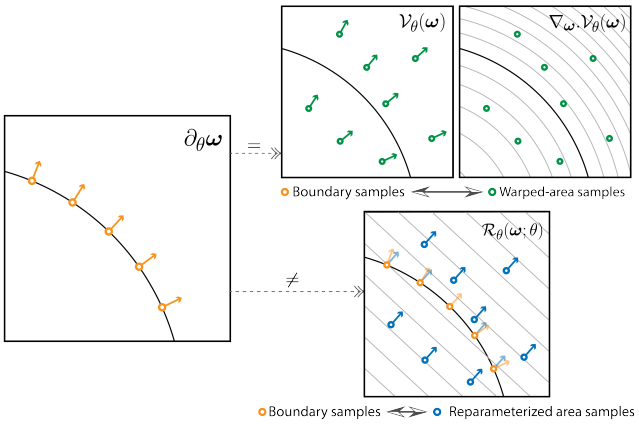


Fig. 4. Warp field formulation. We apply the divergence theorem that shows the equivalence between the boundary integral of Reynolds transport theorem and our area integral. The theorem relates the outgoing flux at the boundary $\partial_\theta \omega$ to the divergence of a warp field $\vec{\mathcal{V}}_\theta(\mathbf{x})$ over the domain. Unlike the reparameterization technique [Loubet et al. 2019]), which uses a uniform rotation to reparameterize the domain, our method produces a spatially varying warp for which this equivalence holds. This introduces a divergence term that intuitively moves the boundary contribution into the interior of the derivative, where it can be computed using standard Monte Carlo rendering.

3.2 Area form of the boundary derivative integral

Eqn. 3 is difficult to evaluate for the rendering integral due to the boundary integral, as mentioned in the introduction. We want to avoid dedicated data structures for edge sampling and efficiently sample discontinuities on a mirror reflection. Furthermore, since path tracing is a recursive integral, evaluating $f(x;\theta)$ in Eqn. 3 requires tracing a full path, making the path tracing cost quadratic to the depth of the path for boundary sampling.

The rendering community has studied efficient techniques for computing the interior integral for decades. It is natural to explore ways to compute the boundary contribution using interior samples.

We derive our solution by directly converting the boundary integral to an area integral, thus achieving consistency and unbiasedness. The key idea is to apply the divergence theorem, a fundamental result in vector calculus.

Divergence theorem (Gauss-Ostrogradsky). Several vector calculus results (Green's theorem, divergence theorem, Stoke's theorem) relate the boundary integral to the interior integral:

$$\oint_{\partial \mathrm{A}} \mathbf{f} \cdot \mathbf{n}\mathrm{d}\mathbf{s} = \iint_{\mathrm{A}-\partial \mathrm{A}} \nabla_\omega \cdot \mathbf{f}\mathrm{d}\omega. \tag{4}$$

In many applications, the divergence theorem and similar vector calculus identities are used for reducing an area integral to a boundary integral since it reduces the dimension of the measure and provides computational benefits (e.g., [Arvo 1995]). On the contrary, to achieve our goal, we need to do the opposite and find an area

integral that is equivalent to the boundary term in Eqn. 3. Furthermore, the Reynolds transport theorem limits the contribution at the boundary to the normal direction. This allows us to ignore the curl term that appears in the more general Stoke's theorem and to only compute the divergence.

We apply the divergence theorem (Eqn. 4) to rewrite the *boundary derivative integral* in the same domain as the *interior derivative integral*:

$$\mathbb{I}_B = \oint_{\partial \mathrm{D}} f(\mathbf{s};\theta)\,(\partial_\theta \mathbf{s}\cdot\hat{\mathbf{n}})\,\mathrm{d}s = \iint_{\mathrm{D}'} \nabla_\omega \cdot \Big(f(\omega;\theta)\vec{\mathcal{V}}_\theta(\omega)\Big)\mathrm{d}\omega$$
$$= \iint_{\mathrm{D}'} \Big(\partial_\omega f(\omega;\theta)\Big)\cdot\vec{\mathcal{V}}_\theta(\omega)\mathrm{d}\omega + \iint_{\mathrm{D}'} \Big(\nabla_\omega\cdot\vec{\mathcal{V}}_\theta(\omega)\Big) f(\omega;\theta)\mathrm{d}\omega, \tag{5}$$

where the *warp field* $\vec{\mathcal{V}}_\theta(\mathbf{x})$ is a *smooth interpolation* of the boundary velocity $\partial_\theta \mathbf{s}$. The last equation is due to the property of the divergence, and we include it for the Monte Carlo estimation we will set up later. Fig. 4 illustrates the geometry of a warp field of a circular boundary.

There are infinitely many different warp fields that satisfy the equation. Following the criteria of the divergence theorem, we can see that the warp field needs to satisfy the following conditions:

(1) *(Continuity)* $\vec{\mathcal{V}}_\theta(\omega)$ must be C_0-continuous for all $x \in \mathrm{D}'$
(2) *(Boundary Consistency)* $\forall \omega_b \in \partial\mathrm{D}$ and $\delta \in \mathbb{R}^+, \exists \epsilon$ such that, $\forall \omega \in \{\omega, |\omega - \omega_b| < \epsilon\}, |\vec{\mathcal{V}}_\theta(\omega) - \partial_\theta \omega_b| < \delta$.

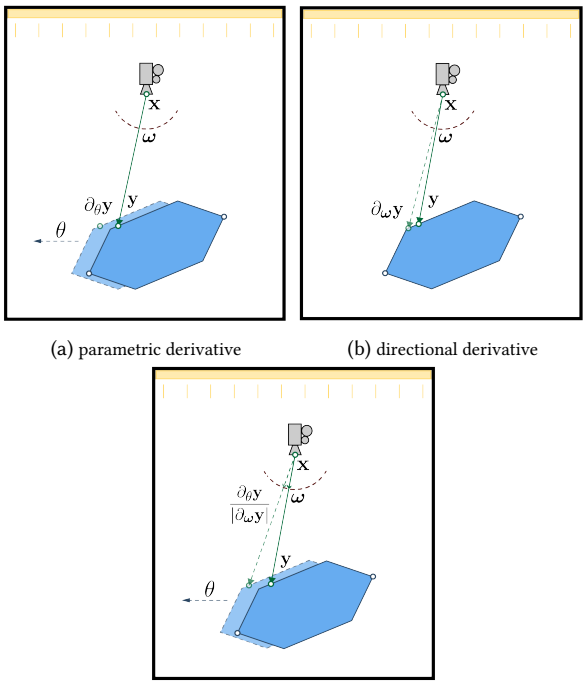
The continuity condition states that the warp field $\vec{\mathcal{V}}_\theta(\mathbf{x})$ must be continuous. The boundary consistency condition states that the warp field must closely match the warp of a boundary point at/near the boundary point. We say a warp field $\vec{\mathcal{V}}_\theta(\mathbf{x})$ is *valid* if it satisfies the above conditions for a given boundary derivative $\partial_\theta \mathbf{x}^{(b)}$.

Intuitively, these conditions can be thought of as a smooth boundary interpolation problem: we want to construct a continuous field, while matching the values at the boundaries. These are crucial conditions that are not trivial to satisfy. In the following subsection, we will carefully identify a family of warp fields $\vec{\mathcal{V}}_\theta(\mathbf{x})$ that satisfies both criteria and is suitable for differentiable rendering. We will also show the connection to Loubet et al. [2019]'s reparameterization method in Section 3.4. Their reparameterization can be seen as constructing a warp field that does not, in general, satisfy the validity criteria, explaining the bias seen in their results.

3.3 Valid warp fields for the rendering integral

The smooth boundary interpolation problem of constructing the warp field $\vec{\mathcal{V}}_\theta(\mathbf{x})$ presents a unique challenge in rendering. In many other fields such as geometry processing or numerical computation, it is often possible to identify all the boundaries a priori, and then discretize the interior in order to construct the smooth field. However, in the area-based rendering formulation, we want to avoid explicit discontinuity enumeration (like the boundary sampling employed in prior work [Li et al. 2018]) in the first place.

More concretely, we are dealing with a *blind interpolation* problem, where we have to construct a valid warp field that can be computed without explicit samples on the boundary.



(a) parametric derivative (b) directional derivative

(c) projecting parametric derivative to solid angle

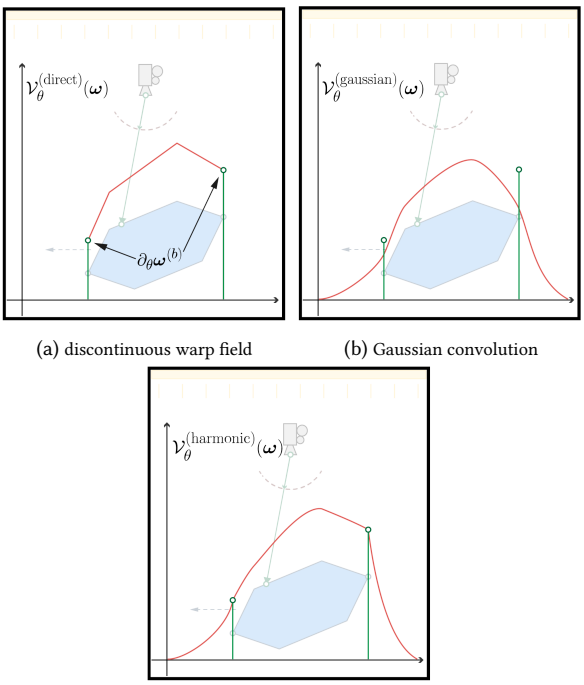
Fig. 5. Projecting the derivative field. **(a)** and **(b)** illustrate the difference between a directional derivative $\partial_{\boldsymbol{\omega}}\mathbf{y}$ and the parameteric derivative $\partial_\theta \mathbf{y}$, since these are important components in our derivation. **(a)** also shows that the parametric derivative is continuous at points on surface $\mathbf{y}$. **(c)** shows the computation of the parametric derivative of a point in solid angle space Ω in terms of the derivatives of the associated scene point $\mathbf{y}$, which we have easy access to. As illustrated, the Jacobian term of the transformation $\boldsymbol{\omega} \to \mathbf{y}$ is used to find the projected version of the parametric derivative.

Exploiting structure in the scene. Our approach exploits the manifold structure of the 3D scene, by observing that the derivative of a 3D scene point $\partial_\theta \mathbf{x}$ is continuous for all surface points $\mathbf{x} \in \mathcal{X}$. The discontinuities in the visibility term only arise when the geometry is projected to the solid angle space $\mathcal{X} \to \Omega$ of the point where the radiance is being evaluated.

Selecting the warp field. We want to define a field $\mathcal{V}_\theta(\boldsymbol{\omega})$ for all $\boldsymbol{\omega} \in \Omega$ such that the continuity and boundary consistency conditions in Section 3.2 are satisfied. Our strategy is to construct a field that satisfies the boundary consistency, but is not necessarily continuous, by differentiating the ray-geometry intersection procedure.

The rendering integral (Eqn. 1) maps a solid angle $\boldsymbol{\omega}$ at position $\mathbf{x}$ to a scene point $\mathbf{y}$ through the ray-scene intersection operator, which we denote by $\mathbf{y} = \textsc{Intersect}(\mathbf{x}, \boldsymbol{\omega}; \theta)$. Specifically, we use the derivatives of the intersection function with respect to the scene parameters θ as our initial (invalid) warp field, by automatically differentiating the intersection function to obtain $\mathbf{y}, \partial_\theta \mathbf{y}, \partial_{\boldsymbol{\omega}} \mathbf{y} = \textsc{Diff-Intersect}(\mathbf{x}, \boldsymbol{\omega}; \theta)$. Concretely, our warp field is:

$$\vec{\mathcal{V}}_\theta^{(\mathrm{direct})}(\boldsymbol{\omega}) = \frac{\partial_\theta \mathbf{y}}{|\partial_{\boldsymbol{\omega}} \mathbf{y}|}. \tag{6}$$



(a) discontinuous warp field (b) Gaussian convolution

(c) our harmonic convolution

Fig. 6. Boundary-aware convolution. **(a)** The form of the warp obtained by using the ray-scene intersection function to transform the domain $\boldsymbol{\omega}$. It is discontinuous at the silhouettes (shown using blue circles) but it is equal to the correct derivative at the boundary (denoted by green lines), **(b)** The warp field produced by convolving the warp field using a Gaussian kernel. This field is continuous and smooth everywhere, but we see that it does not match the true derivative at the boundary. More specifically, in this case the warp at the boundary is an average of the warp on either side of the boundary, only one of which is representative of the warp at the boundary. **(c)** Our proposed convolution method uses inverse distance weights to force the field to match the true warp at the boundary. The resulting warp field is both continuous and consistent at the boundary.

The division by the Jacobian $|\partial_{\boldsymbol{\omega}}\mathbf{y}|$ is for converting the measure of $\partial_\theta \mathbf{y}$ from area measure to solid angle measure. This projection term is the same as the *geometry* term for converting between solid angle and area formulations in path-space rendering methods [Veach 1998]. We use it to project the derivative instead of the radiance.

The warp field $\vec{\mathcal{V}}_\theta^{(\text{direct})}(\boldsymbol{\omega})$ satisfies the boundary consistency criterion, since at the points close to the boundary, the derivative $\frac{\partial_\theta \mathbf{y}}{|\partial_{\boldsymbol{\omega}}\mathbf{y}|}$ approaches the boundary derivative $\partial_\theta \boldsymbol{\omega}_b$.

Intuitively, this states that the rate at which a given point $\boldsymbol{\omega}$ moves is equal to the motion of the corresponding 3D point adjusted by the Jacobian of the projection between the spaces (Fig. 5). Unfortunately, this warp field is *not* valid since it breaks the continuity criterion. For example, consider two angles close together on either side of a boundary, but which intersect different surfaces, and therefore have very different warps.

The discontinuity of the warp field $\vec{\mathcal{V}}_\theta^{(\text{direct})}(\boldsymbol{\omega})$ is also why it is difficult to differentiate the visibility function. In general, for a discontinuity in the visibility function, only one side is representative of the edge. When we try to evaluate the warp at a point on the other side of the discontinuity, the movement of the corresponding 3D point is completely independent of movement of the discontinuity. Still, in order to create a continuous warp, we must somehow ensure our warp field is continuous.

Our solution is to convolve the direct warp field with weights that ensure boundary consistency. This rectifies the underlying discontinuous field $\vec{\mathcal{V}}_\theta^{(\text{direct})}(\boldsymbol{\omega}')$ to produce a smooth and continuous field $\vec{\mathcal{V}}_\theta^{(\text{filtered})}(\boldsymbol{\omega}; w)$

$$\vec{\mathcal{V}}_\theta^{(\text{filtered})}(\boldsymbol{\omega}; w) = \frac{\int_{\Omega'} w(\boldsymbol{\omega}, \boldsymbol{\omega}')\vec{\mathcal{V}}_\theta^{(\text{direct})}(\boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'}{\int_{\Omega'} w(\boldsymbol{\omega}, \boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'}. \tag{7}$$

Next, we discuss how to choose the weights w such that the boundary consistency is preserved.

Boundary-aware convolution. It is not immediately obvious what the weights should be. As a counter example, Fig. 6(b) illustrates a warp field obtained using weights from a normal distribution. The warp field deviates heavily from the true warp at the boundary, and this field is still not valid since it violates boundary consistency. The warp at a point very close to the boundary would be the average of the warp on both sides, which is not, in general, equal to (or even close to) the warp at the boundary.

Our weights need to converge to the derivative at points close to the boundary to produce a valid interpolation. That is, $w(\boldsymbol{\omega}, \boldsymbol{\omega}')$ should grow to infinity when $\boldsymbol{\omega}$ is on the boundary while $\boldsymbol{\omega}'$ approaches $\boldsymbol{\omega}$. We also want the weight to be small when $\boldsymbol{\omega}$ is far from the boundary. We take inspiration from the harmonic interpolation, by selecting weights using the inverse distance to the boundaries.

Unfortunately, we do not know the true distance to the nearest boundary point because it is difficult to find the boundary. Instead we use the concept of a *boundary-test* $\mathcal{B}(\boldsymbol{\omega}')$, which serves as a weaker definition of a boundary. $\mathcal{B}(\boldsymbol{\omega}')$ is essentially a soft indicator function which takes zero value at the boundary and non-negative everywhere else, i.e, $\boldsymbol{\omega}' \in \partial\Omega \implies \mathcal{B}(\boldsymbol{\omega}') = 0$. This is the only key requirement that $\mathcal{B}(\boldsymbol{\omega}')$ needs to satisfy, which gives us great flexibility over its form. Appendix A discusses our boundary test for triangle meshes, while Appendix B discusses additional properties of the boundary test and what would happen with a suboptimal choice of the boundary test.

We can now write down our harmonic weights:

$$w^{(\text{harmonic})}(\boldsymbol{\omega}, \boldsymbol{\omega}') = \frac{1}{\Big(\mathcal{D}(\boldsymbol{\omega}, \boldsymbol{\omega}') + \mathcal{B}(\boldsymbol{\omega}')\Big)} \tag{8}$$

where $\mathcal{D}(\boldsymbol{\omega}, \boldsymbol{\omega}')$ is the distance between $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$. For our implementation, we use the von-Mises Fisher distance $\mathcal{D}(\boldsymbol{\omega}, \boldsymbol{\omega}') = e^{\kappa(1-\langle\boldsymbol{\omega},\boldsymbol{\omega}'\rangle)} - 1$, but any smooth function that satisfies $\mathcal{D}(\boldsymbol{\omega}, \boldsymbol{\omega}) = 0$ can be used.

Our weights satisfy the handy asymptotic property that in the limit where a point $\boldsymbol{\omega}^{(b)}$ approaches a boundary (denoted by $\partial\Omega$),

$\lim_{\boldsymbol{\omega}^{(b)} \to \partial\Omega} \frac{w(\boldsymbol{\omega}^{(b)}, \boldsymbol{\omega})}{\int_{\Omega'} w(\boldsymbol{\omega}^{(b)}, \boldsymbol{\omega}')} = \delta(|\boldsymbol{\omega}^{(b)} - \boldsymbol{\omega}|)$, where $\delta(.)$ is the Dirac delta function. This implies that for a point $\boldsymbol{\omega^{(b)}}$ very close to the boundary, the weights are such that the resulting warp at $\boldsymbol{\omega^{(b)}}$ is equal to the direct warp. As we have already shown, the direct warp is consistent at the boundaries, which implies that the convolution with harmonic weights is also consistent. Fig 6(c) shows the result of our harmonic interpolation.

We only analyze the weights using a limit near the boundary and not the boundary itself, since our warp field only needs to be defined in the smooth interior region $\Omega - \partial\Omega$. This allows our warp field to be undefined at points on the boundary, and the proposed harmonic weights are infinite at such points.

Pixel prefiltering. While, in principle, our method could handle discontinuous pixel prefiltering such as a box filter, we always opt for a continuous filter. For a discontinuous filter, the pixel filter integral $\int_A f(\boldsymbol{\omega}) d\boldsymbol{\omega}$ over the pixel support A needs to take the discontinuity at the support's boundary into account for the divergence theorem to hold. This means our boundary test $B(\boldsymbol{\omega'})$ needs to return 0 at such boundaries. To simplify the boundary test implementation, we always use a Gaussian pixel filter truncated at a radius of 4 times the standard deviation, where the kernel contribution is indistinguishable from the floating point error.

3.4 Relation to the reparameterization method [Loubet et al. 2019]

Before we move to our Monte Carlo sampling algorithm for solving the harmonic convolution integral (Eqn. 7), we discuss the relation of our method to Loubet et al.'s reparameterization.

Although we have derived the area form of the boundary term in a different way, Loubet et al.'s method of transforming the samples using a rotation is actually a special case of our formulation. We show that we can interpret their method as applying a particular warp field that does *not* satisfy the boundary consistency requirement, thus introducing bias to the result.

The reparameterization method applies a transformation $\mathbf{x} = \mathcal{T}(\mathbf{y}; \theta)$ in an attempt to remove the discontinuities of the integral:

$$\mathbb{I} = \int f(\mathbf{x}; \theta) \mathrm{d}\mathbf{x} = \int f(\mathcal{T}(\mathbf{y}; \theta); \theta) \left| \partial_{\mathbf{y}} \mathcal{T}(\mathbf{y}; \theta) \right| \mathrm{d}\mathbf{y} \tag{9}$$

In Appendices C.1 and C.2, we prove that this transformation can be converted to an equivalent warp field $\mathcal{V}_{\boldsymbol{\theta}}(\mathbf{y})$ using the relationship,

$$\mathcal{V}_\theta(\mathbf{x}) = [\partial_{\boldsymbol{\theta}} \mathcal{T}(\mathbf{x}; \boldsymbol{\theta})]_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}. \tag{10}$$

where $\boldsymbol{\theta}_0$ is the point at which we would like to compute the derivative.

Given a warp field, we can also convert it back into a transformation by finding a solution for the right-hand side of Eqn. 10. Since this is a differential equation, there exist many possibilities for $\mathcal{T}(\mathbf{x}; \theta)$'s basic form, depending on the coordinate system. Appendix C.2 discusses linear & rotational solutions to this equation for 2D Euclidean & 3D unit-sphere coordinate systems, respectively. The rotational solution represents the basic transformation that Loubet et al. [2019] uses.

We denote the rotational solutions by $\mathcal{R}(\boldsymbol{\omega}; \theta)$. This limits the set of possible transformations to those with a unit Jacobian $|\partial_{\boldsymbol{\omega}} \mathcal{R}(\boldsymbol{\omega}; \theta)| = 1$, independent of θ. We can deduce that this means the corresponding warp satisfies $\nabla_{\boldsymbol{\omega}} . \mathcal{V}_\theta(\boldsymbol{\omega}) = 0$. Unfortunately, this means that simple rotation-based reparameterization cannot always be a valid warp (Fig. 4 shows an example of a scenario that requires a non-zero divergence).

Loubet et al. recognize this drawback and propose a more complex transformation that samples nearby rays using a von-Mises Fisher distribution (spherical analog of the Gaussian) and constructs a transformation that is the average of these rotations. This can generally be expressed using $\mathcal{R}'(\boldsymbol{\omega}; \theta) = \int k(\boldsymbol{\omega}, \boldsymbol{\omega'}) \mathcal{R}(\boldsymbol{\omega'}; \theta)$. Since the transformation and the warp field are linearly related, the warp field of a convolved transformation $\mathcal{V}'_\theta(\mathbf{x}) = [\partial_\theta \mathcal{R}'(\boldsymbol{\omega}; \theta)]_{\theta=\theta_0}$ is equivalent to the convolution over the warp field of the original transformation $\mathcal{V}'_\theta(\mathbf{x}) = \int k(\boldsymbol{\omega}, \boldsymbol{\omega'}) \mathcal{V}_\theta(\mathbf{x})$. This leads to a scenario similar to that shown in Fig. 6(b) where the resulting transformation is smooth but fails to meet the boundary consistency criterion.

To compensate for this bias, Loubet et al. [2019] introduce a heuristic on top of this convolution. Since the heuristic involves discrete operations such as sorting and comparing object IDs, it is difficult to analytically express the resulting warp field and study its properties. Instead, we turn to empirical comparisons with the ground truth presented in Section 5.

4 MONTE CARLO ESTIMATION OF THE DERIVATIVE

We developed the theory of area sampling and proposed a formulation for the warp field in the previous section. In this section, we develop a Monte Carlo estimator for estimating our divergence area integral (Eqn. 5). In the previous section, we came up with a convolutional warp field that is itself an integral (Eqn. 7). Therefore, we construct a nested Monte Carlo estimator to estimate the derivative integral. We first generate a sample $\boldsymbol{\omega}$ for the outer divergence integral, then generate a set of auxiliary samples $\{\boldsymbol{\omega}'_1 \cdots \boldsymbol{\omega}'_{N'}\}$ to estimate the inner convolution integral at $\boldsymbol{\omega}$. Fig. 7 sums up our rendering process along with a few intra-pixel visualizations to show our intermediate stages.

The convolution integral of the warp field contains a division for normalization. A naïve Monte Carlo estimator of the reciprocal of an integral is not unbiased, since reciprocal is not a linear operation ($E[1/f] \neq 1/E[f]$). Fortunately, it is possible to construct an unbiased Monte Carlo estimator from a consistent estimator using the Russian Roulette de-biasing method [McLeish 2010]. We provide both a consistent estimator and an unbiased estimator in this section.

Finally, naïve Monte Carlo estimation of the harmonic convolution integral can lead to large variance. We discuss variance reduction techniques using control variates.

4.1 Estimating the warp field $\mathcal{V}_\theta(.)$

Our goal is to estimate the divergence area integral (Eqn. 5), whose integrand is $(\nabla_\omega f(\omega; \theta)) \cdot \vec{\mathcal{V}}_\theta(\omega) + \left(\nabla_\omega \cdot \vec{\mathcal{V}}_\theta(\omega)\right) f(\omega; \theta) \mathrm{d}\omega$.

For each direction sample $\boldsymbol{\omega}$ for the outer divergence integral, we sample N' auxiliary directions $\boldsymbol{\omega}'_i$ in order to estimate the warp field $\vec{\mathcal{V}}_\theta(\boldsymbol{\omega})$. Recall that the warp field is a normalized convolution with

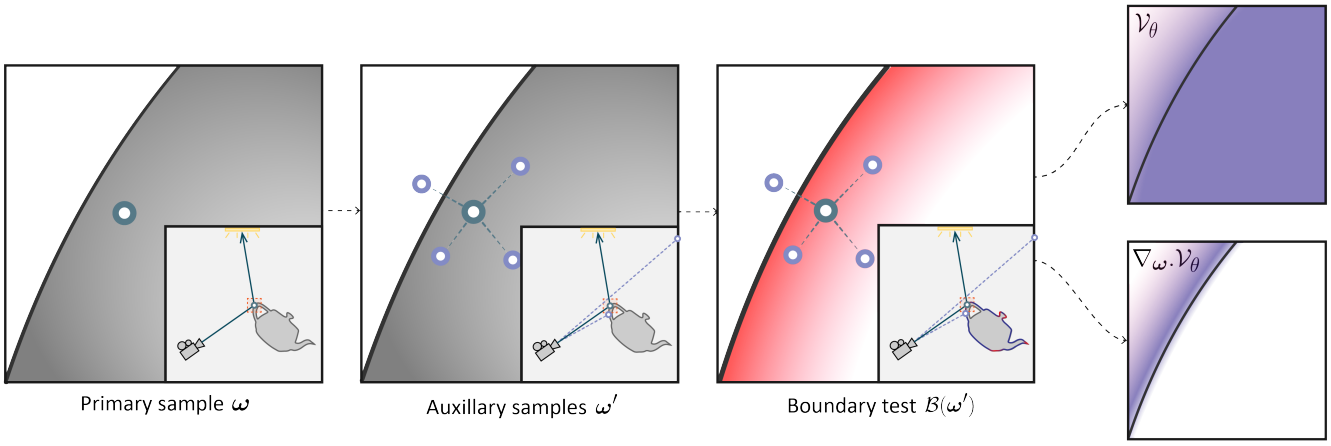


Fig. 7. Our algorithm first samples a ray $\boldsymbol{\omega}$ based on simple path tracing. To compute the boundary contribution to the derivative, we need to estimate the warp function at this point. To achieve this, our method samples a certain number N' of auxiliary rays around this sample $\boldsymbol{\omega}$ using the von-Mises Fisher distribution. We then compute the boundary test at each auxiliary sample $\mathcal{B}(\boldsymbol{\omega}')$ based on surface normals (as described in Appendix A). These boundary values are further processed using Eqn. 8 to produce weights for the samples. Our final step computes the weighted average of the direct warp $\mathcal{V}_\theta^{(\text{direct})}$ at the auxiliary samples to produce estimates for the warp field and its divergence at the primary sample.

Algorithm 1 Monte Carlo estimator of the derivative

function RADIANCE $(\mathbf{x}, \boldsymbol{\omega}_{\text{in}})$
  Sample outgoing radiance direction $\boldsymbol{\omega}$ with incoming direction $\boldsymbol{\omega}_{\text{in}}$
  $\mathbf{y} \leftarrow$ INTERSECT $(\mathbf{x}, \boldsymbol{\omega})$
  $L, \partial_\theta L, \partial_{\boldsymbol{\omega}} L \leftarrow$ RADIANCE $(\mathbf{y}, \boldsymbol{\omega})$
  $\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}), \nabla_{\boldsymbol{\omega}}.\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \leftarrow$ ESTIMATE-WARP $(\boldsymbol{\omega}, \mathbf{y}, N')$
  $\widehat{\partial^b_\theta \mathbb{I}} \leftarrow \langle \partial_{\boldsymbol{\omega}} L, \hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \rangle + L \nabla_{\boldsymbol{\omega}}.\hat{\mathcal{V}}_\theta(\boldsymbol{\omega})$
  $\widehat{\partial_\theta \mathbb{I}} \leftarrow \widehat{\partial^b_\theta \mathbb{I}} + \partial_\theta L$
  $\hat{\mathbb{I}} \leftarrow f_p(\boldsymbol{\omega}_{\text{in}}, \boldsymbol{\omega}) L$
  $\widehat{\partial_{\omega_{\text{in}}} \mathbb{I}} \leftarrow \partial_{\omega_{\text{in}}}(f_p(\boldsymbol{\omega}_{\text{in}}, \boldsymbol{\omega}) L)$
  return $\hat{\mathbb{I}}, \widehat{\partial_\theta \mathbb{I}}, \widehat{\partial_{\omega_{\text{in}}} \mathbb{I}}$
end function

Algorithm 2 Consistent Monte Carlo estimator of the warp field

function ESTIMATE-WARP $(\boldsymbol{\omega}, \mathbf{y}, N')$
  for $i \leftarrow (1 \cdots N')$ **do**
    Sample $\boldsymbol{\omega'}_i$ from vMF distribution with mean $\boldsymbol{\omega}$
    $\mathbf{y}'_i, \partial_\theta \mathbf{y}'_i, \partial_{\boldsymbol{\omega}} \mathbf{y}'_i \leftarrow$ DIFF-INTERSECT $(\mathbf{x}, \boldsymbol{\omega}'_i)$
    $\mathcal{B}_i \leftarrow$ BOUNDARY-DISTANCE $(\mathbf{y}'_i)$
    $w_i \leftarrow \frac{1}{(\exp(\kappa - \kappa\langle \boldsymbol{\omega}, \boldsymbol{\omega}'_i \rangle) - 1) + \mathcal{B}_i} \Big/ \text{PDF}(\boldsymbol{\omega}'_i)$
    $\partial_{\boldsymbol{\omega}} w_i \leftarrow \left(\partial_{\boldsymbol{\omega}} \frac{1}{(\exp(\kappa - \kappa\langle \boldsymbol{\omega}, \boldsymbol{\omega}'_i \rangle) - 1) + \mathcal{B}_i} \right) \Big/ \text{PDF}(\boldsymbol{\omega}'_i)$
    $\mathcal{V}_i^{(1)} \leftarrow \frac{\partial_\theta \mathbf{y}'_i}{|\partial_{\boldsymbol{\omega}} \mathbf{y}'_i|}$
  end for
  $\hat{Z} \leftarrow \sum w_i$
  $\widehat{\partial Z} \leftarrow \sum \partial_{\boldsymbol{\omega}} w_i$
  $\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \leftarrow \frac{\sum \left(w_i \mathcal{V}_i^{(1)} \right)}{\hat{Z}}$
  $\nabla_{\boldsymbol{\omega}}.\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \leftarrow \frac{\sum \langle \partial_{\boldsymbol{\omega}} w_i, \mathcal{V}_i^{(1)} \rangle}{\hat{Z}} - \frac{\sum w_i \langle \mathcal{V}_i^{(1)}, \widehat{\partial Z} \rangle}{\hat{Z}^2}$
  return $\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}), \nabla_{\boldsymbol{\omega}}.\hat{\mathcal{V}}_\theta(\boldsymbol{\omega})$
end function

a kernel of harmonic weights (Eqn. 7 and 8). While the distribution of the harmonic weights depends on the configuration of silhouette edges (Fig. 6), it is also correlated with a normal distribution centered at the outer directional sample $\boldsymbol{\omega}$. Therefore, we importance sample from a normal distribution around our outer directional sample. For hemisphere or sphere sampling, we use the von Mises-Fisher distribution.

Algorithm 1 and 2 detail the nested Monte Carlo estimator that computes the derivative of the radiance in the presence of discontinuities. This estimator is not unbiased for finite N' due to the division over an integral in the harmonic convolution (Eqn. 7), but it is a consistent estimate (it converges to the true derivative as $N' \to \infty$).

In practice, we find that even for fairly complicated scenes, an auxiliary ray count between 4 and 64 provides a very robust estimate of the derivative. However, since there is no closed-form expression for the normalization of the weights $w(\boldsymbol{\omega}, \boldsymbol{\omega'})$, the estimator for the warp field is not unbiased. Still, as $N' \to \infty$, it converges to the ground truth, making it consistent.

De-biasing our estimator. In general, it is possible to produce an unbiased Monte Carlo estimator from a consistent one using Russian roulette [McLeish 2010]. This is a technique commonly used in Bayesian inference [Lyne et al. 2015], nuclear engineering [Booth

Algorithm 3 Unbiased Monte Carlo estimator of the warp field

function ESTIMATE-WARP-RR($\boldsymbol{\omega}$, $\mathbf{y}$, p)
  Draw N' from Geom$(.;p)$
  for $i \leftarrow (1 \cdots N')$ **do**
    Sample $\boldsymbol{\omega'}_i$ from vMF distribution with mean $\boldsymbol{\omega}$
    $\mathbf{y}'_i, \partial_\theta \mathbf{y}'_i, \partial_{\boldsymbol{\omega}} \mathbf{y}'_i \leftarrow$ DIFF-INTERSECT $(\mathbf{x}, \boldsymbol{\omega}'_i)$
    $\mathcal{B}_i \leftarrow$ BOUNDARY-DISTANCE $(\mathbf{y}'_i)$
    $w_i \leftarrow \frac{1}{(\exp(\kappa - \kappa\langle\boldsymbol{\omega}, \boldsymbol{\omega}'_i\rangle) - 1) + \mathcal{B}_i} \Big/ \text{PDF}(\boldsymbol{\omega}'_i)$
    $\partial_{\boldsymbol{\omega}} w_i \leftarrow \left(\partial_{\boldsymbol{\omega}} \frac{1}{(\exp(\kappa - \kappa\langle\boldsymbol{\omega}, \boldsymbol{\omega}'_i\rangle) - 1) + \mathcal{B}_i}\right) \Big/ \text{PDF}(\boldsymbol{\omega}'_i)$
    $\mathcal{V}_i^{(1)} \leftarrow \frac{\partial_\theta \mathbf{y}'_i}{|\partial_{\boldsymbol{\omega}} \mathbf{y}'_i|}$
    $\hat{Z}_i \leftarrow \hat{Z}_{i-1} + w_i$
    $\widehat{\partial Z}_i \leftarrow \partial_{\boldsymbol{\omega}} \hat{Z}_{i-1} + \partial_{\boldsymbol{\omega}} w_i$
    $\hat{T}_i \leftarrow \frac{1}{Z_i} - \frac{1}{Z_{i-1}}$
    $\widehat{\partial T}_i \leftarrow \frac{\widehat{\partial Z}_i}{Z_i^2} - \frac{\widehat{\partial Z}_{i-1}}{Z_{i-1}^2}$
  end for
  $\hat{T} \leftarrow \sum \frac{\hat{T}_i}{\text{Geom}(i;p)}$
  $\widehat{\partial T} \leftarrow \sum \frac{\widehat{\partial T}_i}{\text{Geom}(i;p)}$
  $\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \leftarrow \frac{\sum \left(w_i \mathcal{V}_i^{(1)}\right)}{\hat{Z}}$
  $\nabla_{\boldsymbol{\omega}} . \hat{\mathcal{V}}_\theta(\boldsymbol{\omega}) \leftarrow \sum \langle \partial_{\boldsymbol{\omega}} w_i, \mathcal{V}_i^{(1)} \rangle \hat{T} - \sum w_i \langle \mathcal{V}_i^{(1)}, \widehat{\partial T} \rangle$
  return $\hat{\mathcal{V}}_\theta(\boldsymbol{\omega}), \nabla_{\boldsymbol{\omega}} . \hat{\mathcal{V}}_\theta(\boldsymbol{\omega})$
end function

2007], machine learning [Beatson and Adams 2019], and was recently used for debiasing photon mapping [Qin et al. 2015] and sampling specular light paths [Zeltner et al. 2020].

The intuition of Russian roulette debiasing is to take a converging sequence $T_0, T_1, T_2, \ldots$ produced by a consistent estimator, and construct the following infinite series:

$$\hat{T} = T_0 + \sum_{i=1}^{\infty} (T_i - T_{i-1}). \tag{11}$$

Since $\hat{T}$ converges to the right answer, if we can build an unbiased estimator for $\hat{T}$, we can turn the consistent estimator into an unbiased one. Building unbiased estimators for an infinite series is a common task in physically-based rendering: in path tracing, the infinite Neumann series of indirect bounces is estimated by probabilistically terminating the estimation.

Using the same idea, we sample the series $\hat{T}$ by sampling a finite length from a discrete distribution. We treat N′ as a random variable following a geometric distribution. For every path bounce, during the auxiliary sampling step, we sample a new N' for the estimation of the warp field. This version produces an unbiased estimate of the warp field, which in turn makes our derivative estimate unbiased. Algorithm 3 describes the Monte Carlo estimator.

In practice, using the consistent version with high N' to reduce bias is often faster than Russian Roulette since it can be complicated to manage a dynamic number of rays, especially as rendering infrastructure continues to move to memory-constrained GPUs.

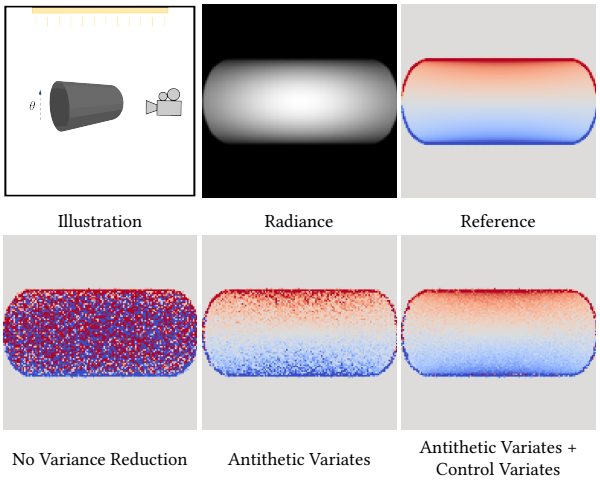


Fig. 8. VARIANCE REDUCTION. We use antithetic variates and control variates to combat the additional variance in interior regions introduced by the kernel convolution. We show the variance reduction from antithetic variates, where we sample on either side of the center of the kernel equally. We further reduce the variance by controlling the estimator with an approximate estimator built using the linear assumption.

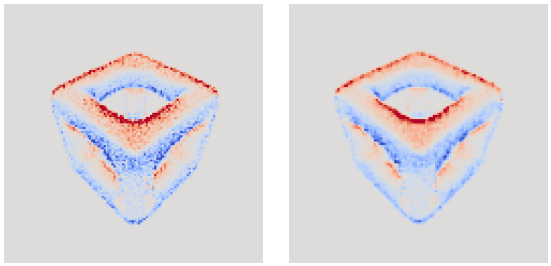


Fig. 9. We compare the gradient variance between our method and the reparameterization method [Loubet et al. 2019]. The comparison uses an equal number of paths per pixel for both methods. Since Loubet et al.'s method traces two correlated paths for each sample, we compare our 32 samples-per-pixel render to their 16 samples-per-pixel render, while maintaining the same number of auxiliary rays (4 per bounce).

4.2 Variance Reduction

Some parts of our estimator exhibit high variance if used directly without explicit variance reduction. We found that smooth regions suffer significantly from the variation of both the harmonic weights and the warp field divergence.

To motivate the variance issue, consider a scene with an infinite, flat emitter where the parameter θ is the emitter's translation. If this emitter is also perfectly normal to our viewing direction (facing the camera), our expected gradient is 0, since all points are moving with exactly the same velocity w.r.t θ. However, we find that the gradient of the weight normalization exhibits very high variance since each of the individual spatial weight derivatives $\nabla_{\boldsymbol{\omega}} w(\boldsymbol{\omega}, \boldsymbol{\omega}')$

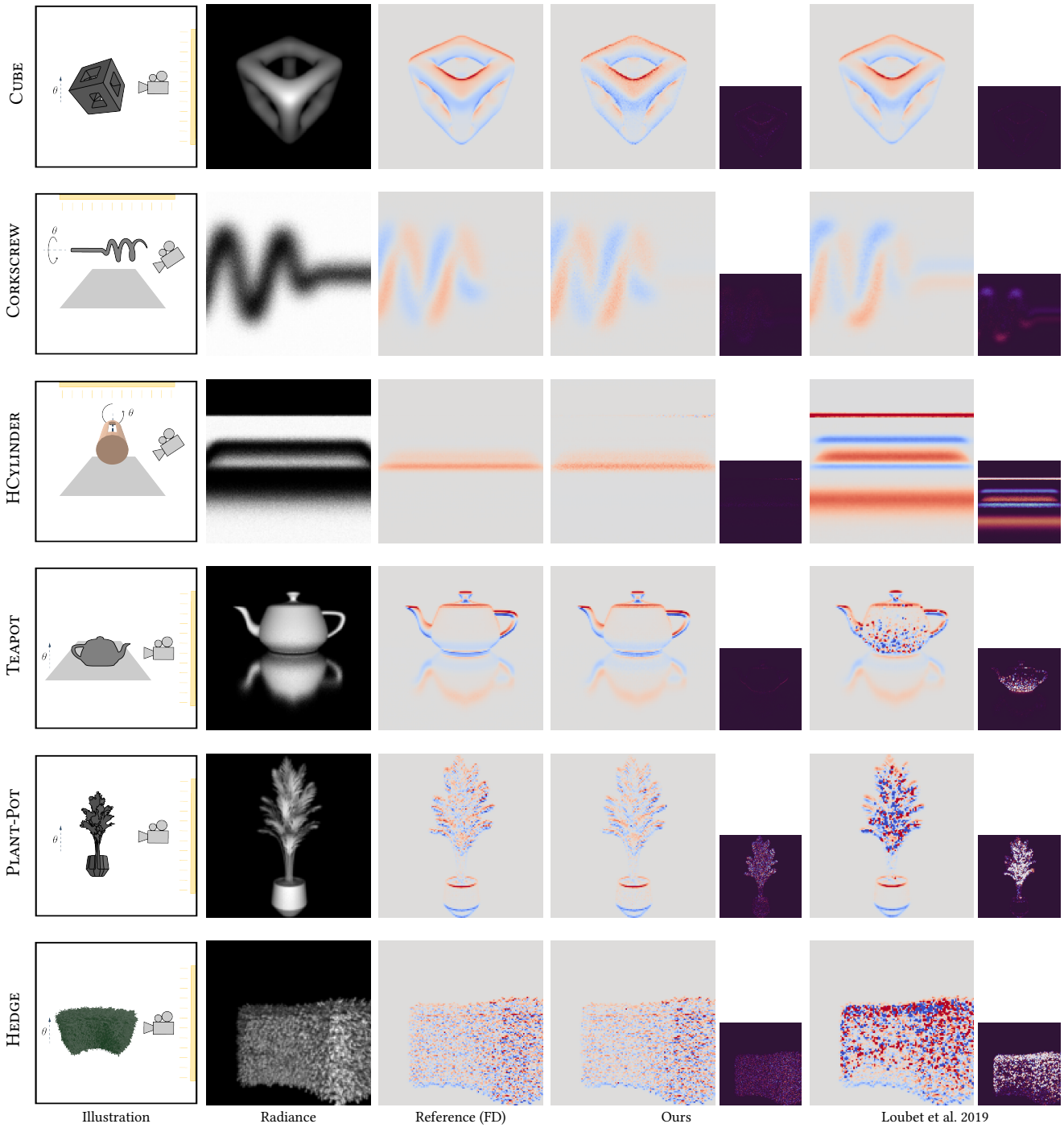


Fig. 10. We compare our per-pixel gradients with the reparameterization method [Loubet et al. 2019] on scenes with varying levels of complexity. Cube has a simple and smooth geometry and motion to show that our area-sampling method produces the correct edge derivative for direct visibility. Corkscrew and HCylinder show rotations of cylinder-like meshes that create difficult situations for area-sampling methods. Teapot shows that our method is accurate even for sharp specular reflections. Plant-Pot and Hedge use meshes with a large number of primitives to produce complicated intra-pixel configurations of edges. Cube, HCylinder and Teapot used a von-Mises Fisher directional light, while the others used rectangular area lights. The insets visualize the absolute error between the finite difference reference and the produced gradient. Our method closely matches the reference in all these cases. These experiments were intended to highlight the bias of each method, and therefore they were compared at high ($\geq$ 128) samples per pixel.

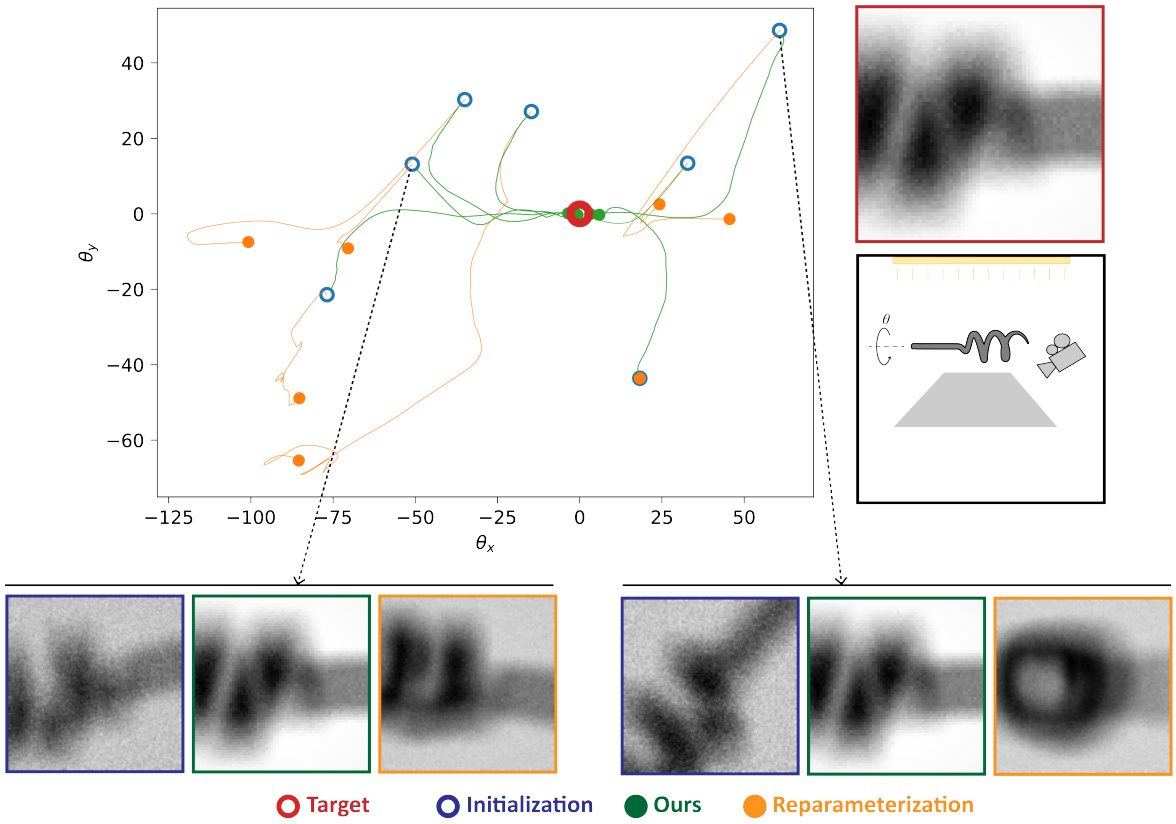


Fig. 11. Optim-Corkscrew: We compare our method with the reparametrization method [Loubet et al. 2019] on a gradient-based optimization task. Both methods are tasked to recover the angular orientation of a corkscrew by observing its soft shadow, starting from the same set of initializations (blue circle). Due to the bias in the reparameterization method, none of the optimizations converged (orange point). In contrast, we are able to recover the target solution from all of the initializations. As recommended by Loubet et al., for reparameterization, we use a *smooth* area light source designed to have a smooth falloff near the boundaries in order to reduce bias. The smooth falloff causes a brightness change to the scene.

can have a large value, depending on the location of the samples w.r.t the center of the distribution. The resulting estimator exhibits high variance even though there is no complexity in the scene, and the expected gradient is 0.

Antithetic Variates. Our approach to counter this variance is to apply *antithetic variates* [Owen 2013]. The key idea behind this widely used variance reduction technique is to transform the samples such that the resulting estimator is *negatively correlated* with the original estimator. It is fairly straightforward to find such a transformation for symmetric weights such as the Gaussian distribution and the von-Mises Fisher distribution. We rotate our samples 180^o about the center of the distribution. Intuitively, the gradient contribution due to the weights at this transformed sample is exactly the negative of the contribution due to the original sample. Fig. 8 demonstrates the significant variance reduction obtained from using antithetic variates.

Control Variates. Antithetic variates offer the most variance reduction when the underlying warp field is roughly *uniform*. Consider the same scene as before, but with a flat emitter at an angle to the camera. Our intersection derivative $\partial_\theta \omega$ is no longer uniform but exhibits approximately linear variation. We find that even though our boundary derivative estimator has an expected value of 0 (since there are no edges, all the contribution comes from the interior derivative integral), it exhibits high variance and therefore requires a lot of samples to converge to the correct value. However, we see that in this particular scenario, we can directly compute the divergence of the warp if we know the slope of the underlying linear surface, without having to go through an extensive estimation step.

Therefore, we use this approximate estimator as a control variate to reduce the variance of our unbiased estimator. Appendix D provides additional details of the construction of the linear approximation and its utility in variance reduction. Fig. 9 compares the

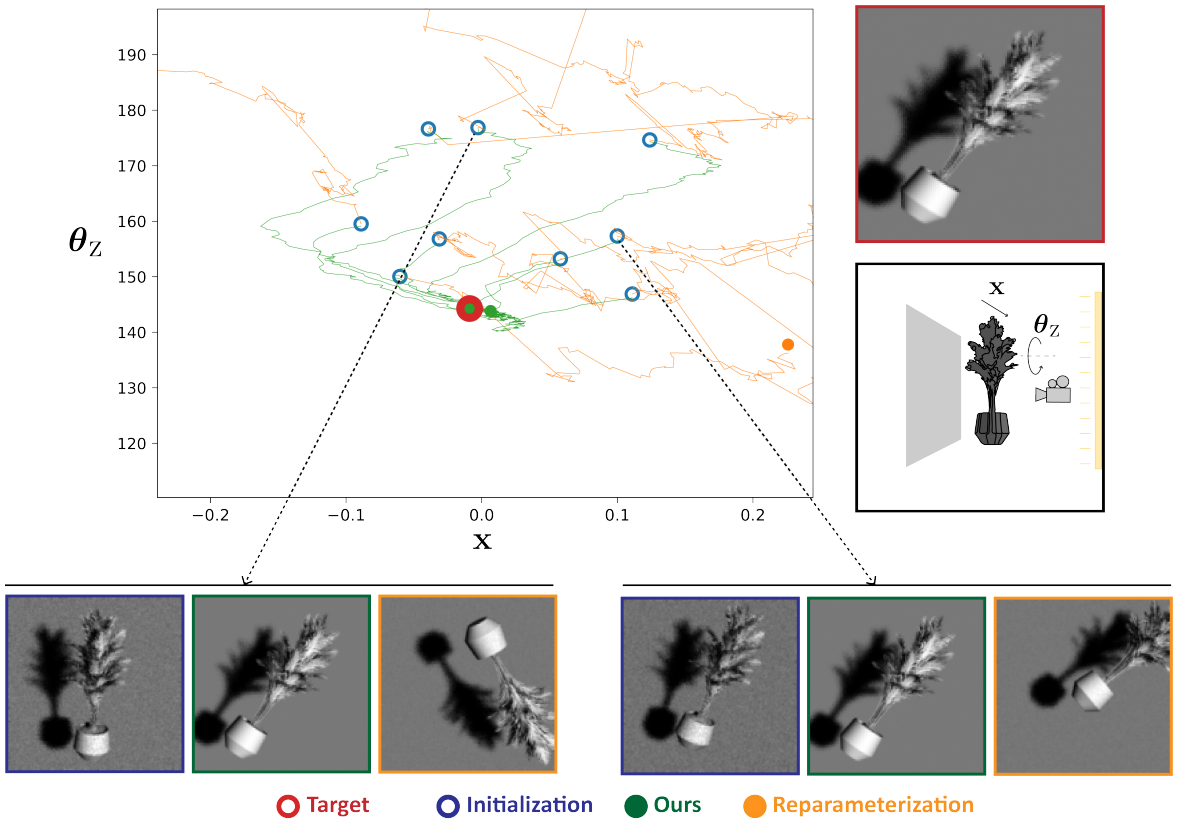


Fig. 12. Optim-Plant: We experiment with 6 degrees of freedom pose optimization of an object with complex structure at 8 samples per pixel. Our estimator provides robust gradients that allow for smooth convergence even at low sample counts. The reparameterization gradient method employs specific heuristics in order to better estimate the correct gradient. However, it was designed with the assumption that a pixel contains a simple configuration of discontinuities. This assumption breaks down here and causes the optimization to diverge completely. We visualize the trajectories for multiple initializations.

relative variance (with antithetic and control variates) and the reparameterization method. The variance reduction methods outlined here are particularly important when used for optimization (Figs. 11 and 12), since we generally use very low sample counts for fast convergence.

5 RESULTS AND DISCUSSION

We implemented both the consistent and unbiased versions of our method on the open-source *redner* repository [Li et al. 2018][2] primarily because it already contains infrastructure to obtain parametric and spatial derivatives ($\partial_\theta L$, $\partial_\omega L$). Our implementation runs on multi-core CPUs and uses Embree [Wald et al. 2014] for ray tracing.

5.1 Comparison with Loubet et al.[2019]

We compare our method to Loubet et al.'s reparameterization method [2019]. Their method relies on approximating a local rotation to compute an approximation to the derivative. Since their method can be interpreted as constructing a warp-field that is not consistent at the boundary, the underlying bias is proportional to the total discrepancy of the warp over the set of discontinuous points (Section 3.4 and Fig. 4). To reduce the bias, Loubet et al. introduced a kernel convolution (similar to our harmonic convolution in Eq. (8)) as well as a heuristic to adjust the warp estimate based on neighborhood sampling. While these adjustments significantly reduce bias for simple configurations of boundaries, it

- fails for scenes with dense and interleaved geometry (such as a dense bush or plant),
- introduces bias when the interior's motion is inconsistent with the boundary and,
- does not easily generalize (more discussion in Section 5.4),

In Fig. 10, we include a comparison of the per-pixel gradients for a single parameter to highlight the systematic bias introduced by the reparameterization method's approximations.

[2]https://github.com/BachiLi/redner

To show the importance of our robust, consistent (or unbiased) gradient, we explore two complex pose estimation problems using a large number of initializations, Optim-Corkscrew (Fig. 11) and Optim-Plant (Fig. 12). We show that the biased reparameterization method consistently fails to obtain the correct solution. In some situations, the method even diverges completely, due to bias introduced by the heuristic.

Both Optim-Corkscrew and Optim-Plant used a target resolution of 128×128, a three-level image pyramid loss and a rectangular area light source. For the reparameterization method, we use the *smooth-area* emitter recommended by Loubet et al. [2019], which blurs the outer 10% of the rectangular light source in order to reduce bias. Optim-Corkscrew used $N = 8$ samples per pixel and $N' = 8$ auxiliary samples, while Optim-Plant used $N = 16$ with $N' = 4$. Both optimizations used the fixed N' variant (consistent warp estimator). The Russian-roulette variant did not provide significant difference in the end result. Optim-Corkscrew using $\kappa = 5 \times 10^5$ for its weight distribution and Optim-Plant using $\kappa = 10^4$. The manually tuned concentration parameter $\kappa = 10^4$ worked very well for all our experiments, providing almost-zero bias, except for the case of the rotating corkscrew, which presents a difficult challenge for area-based methods. This particular case requires a more concentrated von-Mises Fisher distribution to avoid bias when N' is fixed. We leave an adaptive approach to choosing the concentration parameter as future work.

5.2 Comparison with Li et al. [2018]

We also compare both the variance and performance of our algorithm with the unbiased edge-sampling method proposed by Li et al. [2018]. Edge-sampling is a *boundary* method which seeks to find silhouette edges and sample points on them to estimate the boundary integral. Our experiments in Fig. 13 show that while this is very efficient for the first bounce (direct visibility), it struggles to find silhouettes for higher-order effects such as glossy reflections. This effect is worse for scenes with high depth complexity, as shown by the Hedge scene in Fig. 1 and 13, which contains over 200K triangles arranged in dense layers. Since the silhouette sampling is not occlusion-aware, most of the boundary points that are sampled end up occluded. This results in glossy reflections being almost completely missed, manifesting as *fireflies* instead. Edge-sampling is still unbiased, but the poor sampling means it can take a very long time to converge (a similar phenomenon occurs when attempting to render caustics using a simple path tracer). In contrast, our method is *area*-based, and does not suffer from these sampling issues, although this comes at the cost of increased variance in the interior.

5.3 Performance

Despite being CPU-bound, our method has reasonable run-times. For the practical optimization experiments Optim-Corkscrew and Optim-Plant (Fig. 11 and 12), the CPU implementation of our method takes 0.95s and 4.88s respectively, for a full iteration. The GPU-based reparameterization method takes 0.28s and 0.35s per iteration. For our method, we run on an Intel Core i9-9900K. For Loubet et al.'s method [2019], we use their GPU-only implementation running on an 11GB NVIDIA RTX 2080 Ti GPU using OptiX.

Fig. 14 shows that our method has similar or better performance when compared to *redner*'s CPU implementation [Li et al. 2018]. Loubet et al. [2019] show that their performance is similar to *redner* (on GPU). We therefore expect similar performance gains of $\sim 5\times$ if our method is ported to GPU. For simpler geometry, edge-sampling is faster, where our consistent method has run times within a factor of 2. However, with increasing scene complexity, this gap reduces quickly, with Ours-Consistent running faster than Edge-Sampling for the Hedge scene, which contains over 200K primitives.

Fig. 14 also shows the performance of our unbiased Russian roulette variant described in Alg. 3. Even though the average number of auxiliary rays is the same for both Ours-Consistent and Ours-Unbiased, the overhead due to the additional book-keeping required by the Russian roulette technique makes this variant twice as slow. This is one of the reasons we primarily used Ours-Consistent for our optimization experiments.

5.4 Discussion of Generalizability

Unlike existing methods that differentiate in the presence of discontinuities, our method has been derived for a general integral. It relies only on the *boundary test* $\mathcal{B}(.)$, which can be quickly redefined for a different integration problem. The boundary test is, in general, much easier to compute than the actual boundary itself. This opens up an exciting range of possibilities.

Extension to other representations. For defining an easily computable boundary test, we note that triangle meshes, which we use currently, present a harder challenge than many other representations, owing to the locality of information. It is easier to develop an efficient boundary test for alternative representations like signed distance fields, Bezier curves, implicit functions, and most curved geometry (spheres, cylinders, etc.). Thus, in principle, our method could be used with other geometry representations by altering the boundary test. Most existing methods assume a mesh structure composed of planar elements, and it is unclear how to extend them to new representations.

Extension to distribution effects. The derivation of the core idea behind our estimation algorithm is based on the divergence theorem, which can be generalized to higher-dimensional space. Therefore, our warping method can be directly applied to differentiate other rendering domains such as motion blur (3 dimensions with two spatial and one temporal coordinates) and depth-of-field (4 dimensions with two spatial and two angular variables). The boundary test would have to be redefined accordingly to test for boundaries in higher dimensions.

Extension to path space. Our idea can also be potentially applied to more complex domains like path space [Zhang et al. 2020] and primary sample space. However, the definition of the boundaries in path space must be investigated first, since a single point in path space can pass through several occluders. It is not immediately obvious which definition will produce a valid warp field in path space.

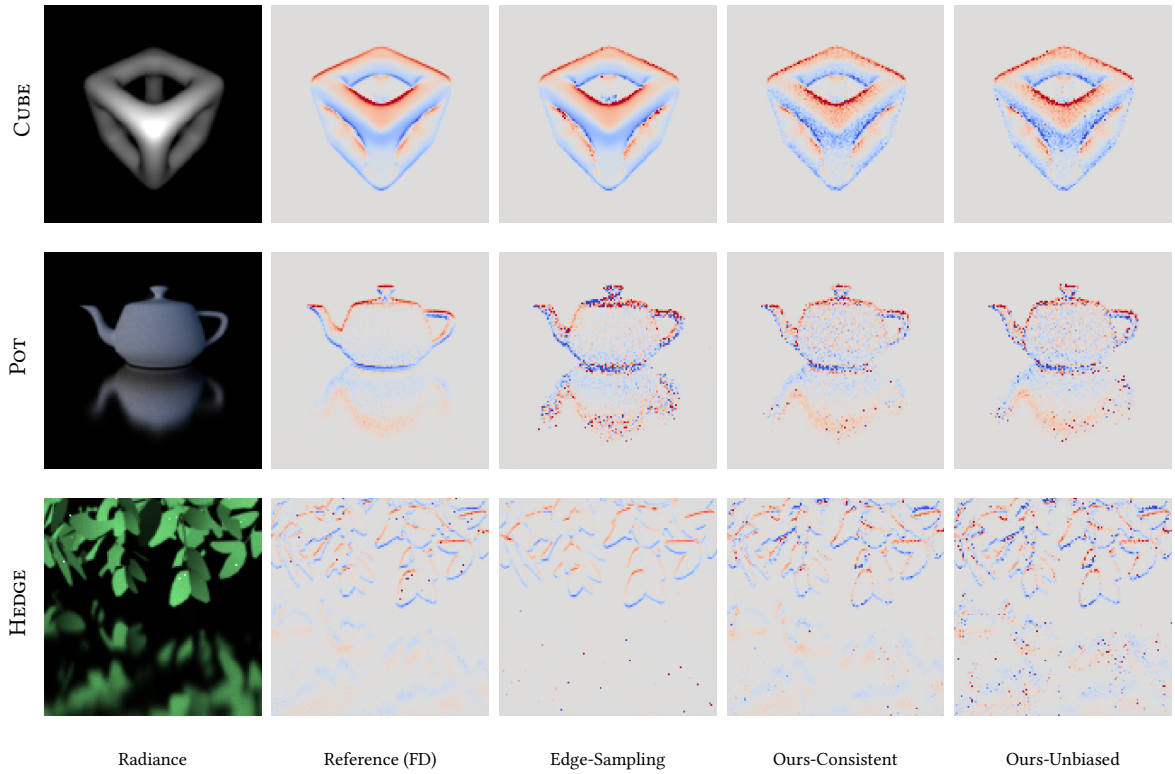


Fig. 13. We compare our per-pixel gradient with the unbiased edge-sampling method proposed by Li et al. [2018]. We perform comparisons with both the consistent and unbiased versions of our algorithm. Since edge-sampling is also unbiased, we focus on variance comparisons by using low sample count gradients. Scene Cube was rendered at $N = 32$ samples per pixel while Pot used $N = 64$ samples per pixel and Hedge used $N = 128$. Ours-Consistent used an auxiliary count of $N' = 8$ while Ours-Unbiased drew the auxiliary count from a *geometric* distribution with the same mean, $N' \sim \text{Geom}(\frac{1}{8})$. Pot used an over-tesselated teapot mesh to show Edge-Sampling's difficulty in handling secondary effects for complex meshes.

5.5 Limitations

Implicit edges. Our current implementation of the boundary test $\mathcal{B}(.)$ does not automatically consider implicit edges (triangle self-intersections) in triangle meshes. The implications of this are subtle: the warp at the implicit boundary point is estimated to be the average over the warp in the neighborhood, since the samples are not correctly weighted. This biased estimate is still usually very accurate, and it produces a result similar to the re-parameterization technique. However, since our boundary test $\mathcal{B}(.)$ does not detect the implicit edges, the resulting warp field has no guarantee of validity, with the consequence that our method is not provably unbiased in the presence of such implicit edges.

Truncation bias in practical implementation. In practice, like other methods that are debiased through Russian Roulette such as path tracing, our method has to truncate at some finite limit, or else run out of time and compute memory. Our choice for the limit ($N' \leq 512$) is large enough to make the resulting truncation bias indistinguishable from floating point error. However, depending on the choice of kernel size and memory limits, we can run into situations where our estimator produces non-negligible bias.

We can reason about this behavior through the Chernoff-Hoeffding theorem, which can be used to find the asymptotic *decay rate* of our inner warp field estimator. The resulting variance and truncation bias from the Russian Roulette modification can be loosely tied to this decay rate. An example of such an analysis for a general Monte Carlo estimator can be found in McLeish's article [2010].

Incoherent workload of Russian roulette debiasing. Our Russian roulette debiasing can introduce an incoherent workload for parallel execution. Each path vertex can have a different amount of auxiliary rays to trace, leading to thread divergence and dynamic memory allocation. We have left an efficient GPU implementation of our method as future work.

6 CONCLUSION

We have used the divergence theorem to unify boundary sampling and area sampling methods in differentiable rendering. We further state the conditions for an area sampling estimator to be consistent/unbiased: continuity of the warp field and matching boundary velocity. We have shown why existing area sampling methods do not produce unbiased derivatives. This allowed us to develop an algorithm that applies an additional convolution using weights inspired by harmonic interpolation to build consistent or unbiased estimators of the derivative. Our method has the simplicity advantage of area sampling, since the warp field can be computed during

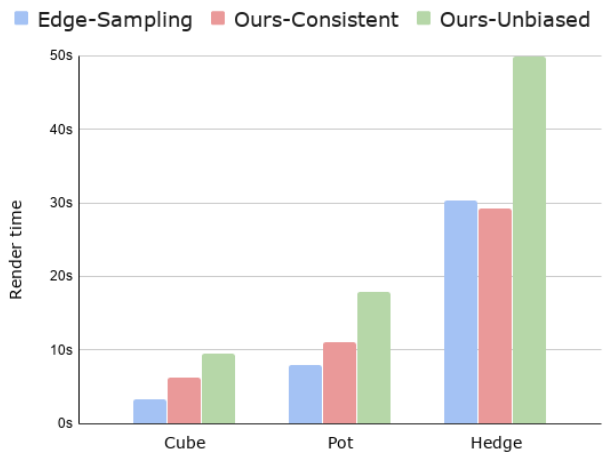


Fig. 14. We compare the render times for the experiments in Fig. 13, which were performed at equal sample count. For edge-sampling, we used the CPU version of *redner*. Across scenes of varying complexity, we see that our method has render times in the same order of magnitude as edge-sampling. The largest disparity is observed for simple scenes where edge-sampling can be twice as fast. As the scene complexity increases, we notice a significant drop in the performance as well as an increase in variance, when compared with our method.

the standard Monte Carlo rendering process, and the advantage of being unbiased. We have demonstrated through inverse rendering experiments that an unbiased estimator of gradients can be critical for stable convergence. Furthermore, area sampling backpropagation methods such as ours are general. They can rely on the sampling infrastructure developed for the forward pass and naturally deal well with depth and geometric complexity.

ACKNOWLEDGMENTS

This work was partially funded by Toyota Research Institute and supported by MIT's Edgerton memorial fellowship. We thank the anonymous reviewers for their helpful comments, and Luke Anderson and Tizian Zeltner for help with proof-reading. We also thank TurboSquid users Oleg_Scott and Zagg3D for the dense hedge geometry and the detailed potted plant geometry, respectively.

REFERENCES

Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance modeling by neural texture synthesis. *ACM Trans. Graph. (Proc. SIGGRAPH)* 35, 4 (2016), 65:1–65:13.

James Arvo. 1994. The Irradiance Jacobian for Partially Occluded Polyhedral Sources. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 343–350.

James Arvo. 1995. Applications of Irradiance Tensors to the Simulation of non-Lambertian Phenomena. In *SIGGRAPH*. 335–342.

Alex Beatson and Ryan P Adams. 2019. Efficient optimization of loops and limits with randomized telescoping sums. In *International Conference on Machine Learning*.

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 187–194.

Thomas E Booth. 2007. Unbiased Monte Carlo estimation of the reciprocal of an integral. *Nuclear science and engineering* 156, 3 (2007), 403–407.

Min Chen and James Arvo. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph.* 19, 4 (2000), 246–278.

Martin de La Gorce, David J Fleet, and Nikos Paragios. 2011. Model-based 3D hand pose estimation from monocular video. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 9 (2011), 1793–1805.

Harley Flanders. 1973. Differentiation under the integral sign. *The American Mathematical Monthly* 80, 6 (1973), 615–627.

Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. 2018. Unsupervised Training for 3D Morphable Model Regression. In *Computer Vision and Pattern Recognition*.

Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. 2015. Improved Half Vector Space Light Transport. *Comput. Graph. Forum (Proc. EGSR)* 34, 4 (2015), 65–74.

Aaron Hertzmann and Denis Zorin. 2000. Illustrating smooth surfaces. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 517–526.

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 (2012), 58:1–58:13.

André Jalobeanu, Frank O Kuehnel, and John C Stutz. 2004. Modeling images of natural 3d surfaces: Overview and potential applications. In *Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, 188–188.

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

Anton S Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4 (2014), 102:1–102:13.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *Computer Vision and Pattern Recognition*. IEEE, 3907–3916.

Tzu-Mao Li. 2019. *Differentiable Visual Computing*. Ph.D. Dissertation. Massachusetts Institute of Technology. Advisor(s) Durand, Frédo.

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.

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 34, 6 (2015), 209:1–209:13.

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2019. Inverse Rendering for Complex Indoor Scenes: Shape, Spatially-Varying Lighting and SVBRDF from a Single Image. *arXiv preprint arXiv:1905.02722* (2019).

Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. 2017. Material Editing Using a Physically Based Rendering Network. In *International Conference on Computer Vision*. IEEE, 2280–2288.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *International Conference on Computer Vision* (2019).

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *European Conference on Computer Vision*, Vol. 8695. ACM, 154–169.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (2019), 228.

Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Langevin Monte Carlo Rendering with Gradient-based Adaptation. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020).

Anne-Marie Lyne, Mark Girolami, Yves Atchadé, Heiko Strathmann, Daniel Simpson, et al. 2015. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science* 30, 4 (2015), 443–467.

J.R. Magnus and H. Neudecker. 1999. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley. https://books.google.com/books?id=0CXXdKKiIpQC

Don McLeish. 2010. A general method for debiasing a Monte Carlo estimator. *Monte Carlo Methods and Applications* 17 (2010), 301 – 315.

Don Mitchell and Pat Hanrahan. 1992. Illumination from curved reflectors. (1992), 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 (July 2020). https://doi.org/10.1145/3386569.3392406

Matt Olson and Hao Zhang. 2006. Silhouette extraction in Hough space. *Comput. Graph. Forum (Proc. Eurographics)* 25, 3 (2006), 273–282.

Art B. Owen. 2013. Monte Carlo theory, methods and examples.

Gustavo Patow and Xavier Pueyo. 2003. A survey of inverse rendering problems. *Comput. Graph. Forum* 22, 4 (2003), 663–687.

Hao Qin, Xin Sun, Qiming Hou, Baining Guo, and Kun Zhou. 2015. Unbiased Photon Gathering for Light Transport Simulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 34, 6 (2015).

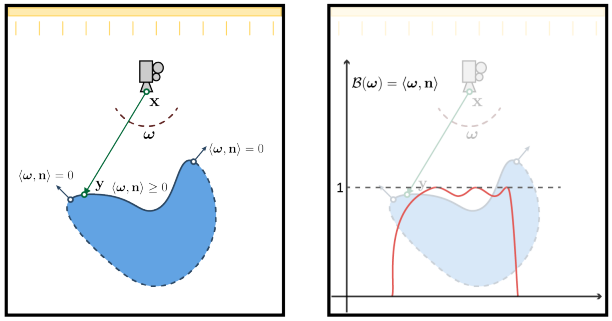
Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A First-order Analysis of Lighting, Shading, and Shadows. *ACM Trans. Graph.* 26, 1 (2007), 2.
Osborne Reynolds, Arthur William Brightmore, and William Henry Moorby. 1903. *The sub-mechanics of the universe.* Vol. 3. University Press.
Helge 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 *International Conference on Computer Vision.* IEEE, 765–773.
Damien Rioux-Lavoie, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai. 2020. Delayed Rejection Metropolis Light Transport. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 3 (April 2020). https://doi.org/10.1145/3388538
Maxime Roger, Stéphane Blanco, Mouna El Hafi, and Richard Fournier. 2005. Monte Carlo estimates of domain-deformation sensitivities. *Physical review letters* 95, 18 (2005), 180601.
Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. 2000. Silhouette Clipping. In *SIGGRAPH.* ACM Press/Addison-Wesley Publishing Co., 327–334.
Mikio Shinya, T. Takahashi, and Seiichiro Naito. 1987. Principles and Applications of Pencil Tracing. *Comput. Graph. (Proc. SIGGRAPH)* 21, 4 (1987), 45–54.
Vadim N Smelyansky, Robin D Morris, Frank O Kuehnel, David A Maluf, and Peter Cheeseman. 2002. Dramatic improvements to feature based stereo. In *European Conference on Computer Vision.* Springer, 247–261.
Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph.D. Dissertation. Stanford University. Advisor(s) Guibas, Leonidas J.
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. (Proc. SIGGRAPH)* 33, 4 (2014), 143:1–143:8.
Greg Ward and Paul Heckbert. 1992. Irradiance Gradients. In *Eurographics Workshop on Rendering.* Eurographics Association, 85–98.
Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. 2020. Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020).
Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular manifold sampling for rendering high-frequency caustics and glints. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020), 149.
Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 6 (2020), 143:1–143:19.
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 (2019), 227.

A BOUNDARY-TEST $\mathcal{B}(.)$

Our implementation of the boundary-test function $\mathcal{B}(.)$ takes advantage of the loose requirements defined in Section 3.3. The primary requirement is that $\mathcal{B}(\boldsymbol{\omega}) \to 0$ as $\boldsymbol{\omega}$ approaches the boundary. As Fig. 15 illustrates, we can take advantage of the surface normals to design a good boundary test. The dot product between the surface normal and the ray direction is 0 at silhouette points, and it is a continuous function within a boundary ($\mathcal{B}(\boldsymbol{\omega})$ does not need to be continuous across boundaries). Since this inner product can be computed quickly for various representations, it serves as a light-weight starting point for computing $\mathcal{B}(\boldsymbol{\omega})$. For triangle meshes, we need to handle open edges that only associate with one face. We additionally control the spread by applying a nonlinear function to the inner-product at each vertex.

We compute the boundary test $\mathcal{B}(.)$ for triangle meshes using the following procedure

(1) We find the triangle that intersects the ray in the direction $\boldsymbol{\omega}$. For each triangle vertex, check if any of the adjacent edges are silhouette edges. An edge is *silhouette* if it (i) has an adjacent face that is back-facing, or (ii) it is open (has only one adjacent face).
(2) Compute a value $\mathcal{B}_v$ for each vertex v. If a vertex v is associated with a silhouette edge, set $\mathcal{B}_v = 0$. For vertices with no silhouette edges, we compute the non-linear composition of the dot product $\langle \boldsymbol{\omega}, \mathbf{n} \rangle$, $\mathcal{B}_v = \frac{1-(1-\langle \boldsymbol{\omega}, \mathbf{n} \rangle^2)}{1-(1-\beta)(1-\langle \boldsymbol{\omega}, \mathbf{n} \rangle^2)}$. Here, $\mathbf{n}$ is the normal at each vertex, pre-computed as the average normal of all adjacent faces. The parameter β controls the spread-rate and we use $\beta = 0.01$ for our experiments. We found empirically that this value gave us the least variance and was not particularly sensitive to scene composition or geometry.
(3) We use barycentric coordinates to propagate the $\mathcal{B}(.)$ at the vertices to the intersection point.



(a) (b)

Fig. 15. **(a)** An illustration of a complex smooth surface. We note that all points visible to the camera are such that the inner product between the camera direction and the normal direction $\langle \boldsymbol{\omega}, \mathbf{n} \rangle$ is positive. At the silhouettes, this inner product is 0. **(b)** shows the variation of the dot product over the surface, demonstrating that within the interior (excluding boundary points), this is a smooth function.

However, for some over-tessellated objects with low geometric detail, the mesh elements constituting the boundary can become so small that the resulting boundary function is very sparse, leading to a noisy estimator. It is possible to design boundary tests with better behavior by propagating information on the mesh, although this can increase the complexity.

B CONTINUITY OF THE WARP FUNCTION

In Section 3.3 we defined a set of weights based on harmonic interpolation (Eqn. 8). The harmonic weights depend directly on the boundary-test function $\mathcal{B}(.)$. Here, we discuss some additional properties of $\mathcal{B}(.)$.

We can reason about the continuity of the $\mathcal{V}_\theta$ through its divergence. If the divergence is finite for interior (non-boundary) points, then it must follow that the warp field is continuous in the interior. Using typical vector calculus identities and the product rule of

differentiation, we represent the divergence of $\mathcal{V}_\theta$ as

$$\nabla_{\boldsymbol{\omega}}.\mathcal{V}_\theta(\boldsymbol{\omega}) = \int_\Omega \frac{\partial_{\boldsymbol{\omega}} w(\boldsymbol{\omega},\boldsymbol{\omega}')^T \mathcal{V}_\theta^{(d)}(\boldsymbol{\omega}') \int_\Omega w(\boldsymbol{\omega},\boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'}{\left(\int_\Omega w(\boldsymbol{\omega},\boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'\right)^2} - \frac{w(\boldsymbol{\omega},\boldsymbol{\omega}')\mathcal{V}_\theta^{(d)}(\boldsymbol{\omega}')^T \int_\Omega \partial_{\boldsymbol{\omega}} w(\boldsymbol{\omega},\boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'}{\left(\int_\Omega w(\boldsymbol{\omega},\boldsymbol{\omega}')\mathrm{d}\boldsymbol{\omega}'\right)^2}\mathrm{d}\boldsymbol{\omega}' \quad (12)$$

.

This divergence equation, aside from being an important quantity for our algorithm, also only depends on the derivative of the weights w.r.t the *primary sample* $\boldsymbol{\omega}$. As long as this derivative exists at all points in the interior, the divergence is finite. It is easy to see that our proposed harmonic weights in Eqn. 8 satisfy this property as long as our distance function $\mathcal{D}(\omega,\omega')$ is differentiable. This means that $\mathcal{B}(.)$ does not need to be differentiable or even continuous for the resulting warp field to be continuous.

C RELATION TO REPARAMETERIZATION

Extending the discussion in Section 3.4, we prove that a warp field $\mathcal{V}_\theta(\mathbf{x})$ can be converted into a reparameterization $\mathbf{x} = \mathcal{T}(\mathbf{y};\theta)$, and vice-versa. Throughout this proof, we assume that we are attempting to find the derivative w.r.t θ at a known point in parameter space, which we will denote as θ_0.

C.1 $\mathcal{T}(\mathbf{x};\theta) \rightarrow \mathcal{V}_\theta(\mathbf{x})$

For a given reparameterization $\mathcal{T}(\mathbf{y};\theta)$, the resulting integral can be re-written as

$$\mathbb{I} = \int f(x)\mathrm{d}\mathbf{x} = \int f(\mathcal{T}(\mathbf{y};\theta);\theta)|\mathrm{det}J_\mathcal{T}|\mathrm{d}\mathbf{y}$$

where $J_\mathcal{T}$ is the Jacobian of the reparametrization.

The resulting parametric derivative at $\theta = \theta_0$ is then

$$\partial_\theta\mathbb{I} = \int \left[\partial_\theta f(\mathcal{T}(\mathbf{y};\theta);\theta)|\mathrm{det}J_\mathcal{T}|\right]_{\theta=\theta_0}\mathrm{d}\mathbf{y} + \int \left[f(\mathcal{T}(\mathbf{y};\theta);\theta)\partial_\theta|\mathrm{det}J_\mathcal{T}|\right]_{\theta=\theta_0}\mathrm{d}\mathbf{y}$$

We use two properties to obtain the final form of the derivative:

(1) By construction, the transformation $\mathcal{T}(\mathbf{y};\theta)$, obeys the rule $\theta = \theta_0, \mathcal{T}(\mathbf{y};\theta_0) = \mathbf{y}$ [Loubet et al. 2019]
(2) The Jacobi's formula [Magnus and Neudecker 1999], which states that $\partial_\theta|\mathrm{det}J_\mathcal{T}| = \mathrm{tr}(\mathrm{adj}(J_\mathcal{T})\partial_\theta J_\mathcal{T})$

The first property also implies that the adjugate of the Jacobian at $\theta = \theta_0$ is I, the identity matrix. We also note that the trace of a square Jacobian matrix is simply the divergence of the underlying mapping.

Applying these observations, we arrive at the following equation:

$$[\partial_\theta\mathbb{I}]_{\theta=\theta_0} = \int \partial_\theta f(\mathbf{y};\theta_0)\mathrm{d}\mathbf{y} + \int \partial_\mathbf{y} f(\mathbf{y};\theta_0)[\partial_\theta\mathcal{T}(\mathbf{y};\theta)]_{\theta=\theta_0}\mathrm{d}\mathbf{y} + \int f(\mathbf{y};\theta_0)\nabla_\mathbf{y}.[\partial_\theta\mathcal{T}(\mathbf{y};\theta)]_{\theta=\theta_0}\mathrm{d}\mathbf{y}$$

.

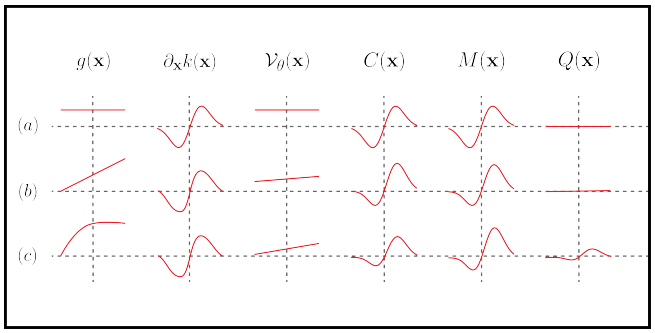


Fig. 16. We illustrate of how our control variates estimator reduces the variance in 3 increasingly complex examples. The integrand we want to estimate is $M(\mathbf{x}) = g(\mathbf{x})\partial_\mathbf{x}k(\mathbf{x})\mathcal{V}_\theta(\mathbf{x})$. Our control variate is $C(\mathbf{x})$. **(a)** and **(b)** show scenarios roughly uniform and roughly linear functions for the underlying radiance and warp field. Our linear approximation-based estimator $C(\mathbf{x})$ provides a near-perfect correlation in both cases, almost completely removing the variance of the resulting estimator $Q = M - C + \mathbb{E}[C]$. In **(c)**, we see a more complex underlying function and while our approximate estimator does not capture all the features, it is correlated enough to provide significant variance reduction.

This equation is now in the form of Eqn. 5, which implies that using a warp field $\mathcal{V}_\theta(\mathbf{x}) = [\partial_\theta\mathcal{T}(\mathbf{x};\theta)]_{\theta=\theta_0}$ produces an equivalent integral.

C.2 $\mathcal{V}_\theta(\mathbf{x}) \rightarrow \mathcal{T}(\mathbf{x};\theta)$

To convert back from a warp field to a transformation, we find solutions to the differential equation $\mathcal{V}_\theta(\mathbf{x}) = [\partial_\theta\mathcal{T}(\mathbf{x};\theta)]_{\theta=\theta_0}$. There are several possibilities for a transformation that produces the same warp field. Here, we detail the simplest warp fields for two different coordinate systems.

2D free coordinates. Consider the transformation

$$\mathcal{T}(\mathbf{y};\theta) = \mathbf{y} + (\theta - \theta_0)\mathcal{V}_\theta(\mathbf{x}) \quad (13)$$

This satisfies property 1 from Appendix C.1, i.e., $\partial_\mathbf{y}\mathcal{T}(\mathbf{y};\theta) = I$. Also, note that $[\partial_\theta\mathcal{T}(\mathbf{y};\theta)]_{\theta=\theta_0} = \mathcal{V}_\theta(\mathbf{x})$.

The result in Appendix C.1 can now be applied to this transformation, which shows that the reparameterized integral that is equivalent to using our warp field integral with the given warp field $\mathcal{V}_\theta(\mathbf{y})$.

3D unit-sphere coordinates. In a similar way, we can find a simple transformation for coordinates that are constrained on the unit sphere. A simple translation would be incorrect here since it would transform points off the spherical domain. Instead, the transformation will have to be a rotation.

$$\mathcal{R}(\boldsymbol{\omega};\theta) = \cos\left((\theta - \theta_0) * |\mathcal{V}_\theta(\boldsymbol{\omega})|\right)\boldsymbol{\omega} + \sin\left((\theta - \theta_0) * |\mathcal{V}_\theta(\boldsymbol{\omega})|\right)\hat{\mathcal{V}}_\theta(\boldsymbol{\omega})$$

D CONTROL ESTIMATOR

In this section, we elaborate upon the discussion in Section 4.2. More specifically, we provide an explicit form for the linear approximation used to reduce the variance of our main estimator through the method of control variates.

We revisit the warped-area boundary integral (Eqn. 5) in order to understand the source of the variance:

$$\mathbb{I}_B = \iint_{\mathrm{D}'} \left(\nabla_{\mathbf{x}} f(\mathbf{x};\theta)\right) \cdot \vec{\mathcal{V}}_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} + \iint_{\mathrm{D}'} \left(\nabla_{\mathbf{x}} \cdot \vec{\mathcal{V}}_\theta(\mathbf{x})\right) f(\mathbf{x};\theta) \mathrm{d}\mathbf{x}$$

We use the pixel prefilter integral to exemplify our control variates. Our control variates is applicable to secondary bounces as well. When computing derivatives for an entire image, each pixel is an integral over the effective radiance $f(\mathbf{x};\theta)$, which is the product of a response function (the pre-filter $k(\mathbf{x})$) and the incoming radiance function ($g(\mathbf{x};\theta)$). In all our experiments, the pre-filter is a standard normal distribution with $\sigma = 1/2$px. Here, $\mathbf{x}$ is a coordinate over the 2D image plane.

For clarity, we will express our integrals in terms of random variables and expectations. We will also use linear algebra notation. The first integral in the boundary integral formulation can now be written in terms of a two-dimensional uniform random variable $\mathbf{x}$ over the pixel's domain (We assume that $\mathbf{x} = \vec{0}$ is the pixel center).

$$\mathbb{I}_B^{(1)} = \mathbb{E}_{\mathbf{x}}\left[g(\mathbf{x};\theta)\partial_{\mathbf{x}} k(\mathbf{x}) \cdot \vec{\mathcal{V}}_\theta(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x})\partial_{\mathbf{x}} g(\mathbf{x};\theta) \cdot \vec{\mathcal{V}}_\theta(\mathbf{x})\right]$$

The primary source of variance is the first term of this expansion, $M = g(\mathbf{x};\theta)\partial_{\mathbf{x}} k(\mathbf{x}) \cdot \vec{\mathcal{V}}_\theta(\mathbf{x})$. The term $\partial_{\mathbf{x}} k(\mathbf{x})$ is just the derivative of the normal distribution, the expected value of which is 0. However, in general, this term is non-zero across the domain of the pixel. While we know the expected value of $\partial_{\mathbf{x}} k(\mathbf{x})$, the product $\mathcal{V}_\theta(\mathbf{x}) g(\mathbf{x};\theta)\partial_{\mathbf{x}} k(\mathbf{x})$ has an unknown mean value.

To circumvent this, we construct an approximate estimator by finding a linear approximation for $g * \mathcal{V}$ such that the resulting estimator has a known (i.e., analytically computable) mean value. A locally linear approximation can be constructed using the mean values and the mean first-order derivatives of $g(.)$ ($\mu_g = \mathbb{E}_{\mathbf{x}}[g]$, $\nabla_g = \mathbb{E}_{\mathbf{x}}[\partial_{\mathbf{x}} g]$) and $\mathcal{V}(.)$ ($\mu_{\mathcal{V}} = \mathbb{E}_{\mathbf{x}}[\mathcal{V}]$, $\nabla_{\mathcal{V}} = \mathbb{E}_{\mathbf{x}}[\partial_{\mathbf{x}} \mathcal{V}]$). Fortunately, estimating these quantities is easy since we have access to the derivatives of the radiance $g(.)$ as well as the warp field $\mathcal{V}(.)$ when computing the primary estimate.

The effective first order derivative of the approximation is then the matrix $A = \mu_g \nabla_{\mathcal{V}} + \mu_{\mathcal{V}} \nabla_g^T$. Our control variate is then (in the quadratic form)

$$\mathrm{C} = \partial_{\mathbf{x}} k^T A \mathbf{x} \tag{14}$$

Since $g(.)$ is a scalar and $\mathcal{V}(.)$ is a vector, the first-order derivatives of these quantities are a vector and a matrix, respectively. The remaining random variables $k(\mathbf{x})$ and $\mathbf{x}$ have known distributions. Therefore the expected value can be computed by using some common statistics results, specifically the expected value of a quadratic form:

$$\mathbb{E}[\mathrm{C}] = \mathrm{tr}\Big(A \operatorname{cov}(\partial_{\mathbf{x}} k^T, \mathbf{x})\Big) + \mathbb{E}[\partial_{\mathbf{x}} k]^T A \, \mathbb{E}[\mathbf{x}]$$

For the case of the normal distribution, this simplifies to

$$\mathbb{E}[\mathrm{C}] = \sigma^2 \mathrm{tr}(A)$$

Fig. 16 illustrates how C is, in general, correlated with M and therefore the resulting estimator $Q = M - C + \mathbb{E}[C]$ has reduced variance.