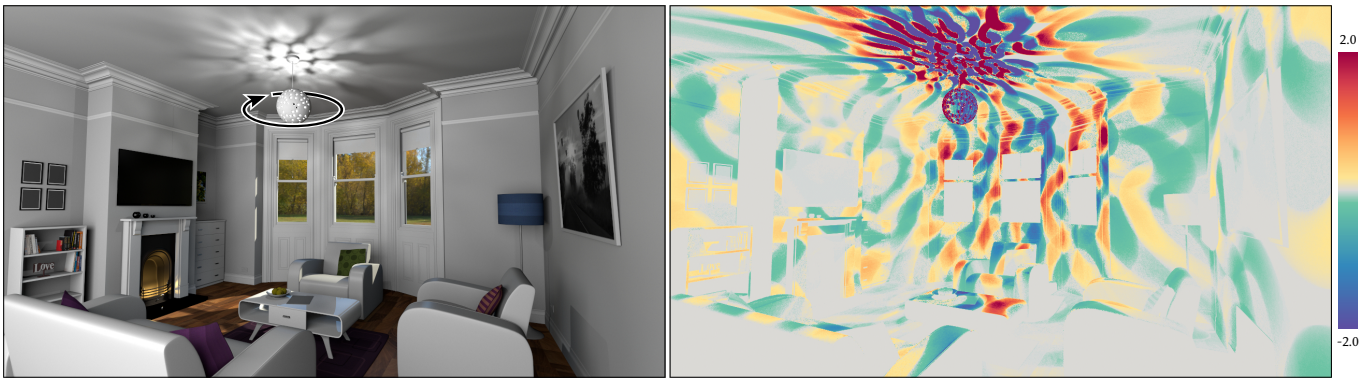
Projective Sampling for Differentiable Rendering of Geometry

ZIYI ZHANG, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
NICOLAS ROUSSEL, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland



(**a**) Primal rendering and perturbation of lamp rotation (**b**) Forward derivative computed using projective sampling

Fig. 1. The visibility function plays a crucial role in differentiable rendering of geometry. Parameter changes that influence visibility (e.g., a rotation of the light source in **(a)** can generate a significant derivative contribution shown in (**b**) that is difficult to sample using currently existing methods. We project path segments generated by primal rendering algorithms (e.g., direct illumination sampling) onto nearby silhouettes and organize them in a uniform or adaptive guiding data structure to enhance numerical integration of the challenging boundary term. Compared to prior work [Zhang et al. 2020], uniform guiding cuts errors (RMSE) by an average of 8.1×, and adaptive guiding yields an additional 2.7× improvement.

Discontinuous visibility changes at object boundaries remain a persistent source of difficulty in the area of differentiable rendering. Left untreated, they bias computed gradients so severely that even basic optimization tasks fail.

Prior path-space methods addressed this bias by decoupling boundaries from the interior, allowing each part to be handled using specialized Monte Carlo sampling strategies. While conceptually powerful, the full potential of this idea remains unrealized since existing methods often fail to adequately sample the boundary proportional to its contribution.

This paper presents theoretical and algorithmic contributions. On the theoretical side, we transform the boundary derivative into a remarkably simple local integral that invites present and future developments.

Building on this result, we propose a new strategy that projects ordinary samples produced during forward rendering onto nearby boundaries. The resulting projections establish a variance-reducing guiding distribution that accelerates convergence of the subsequent differential phase.

We demonstrate the superior efficiency and versatility of our method across a variety of shape representations, including triangle meshes, implicitly defined surfaces, and cylindrical fibers based on Bézier curves.

Authors' addresses: Ziyi Zhang, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, ziyi.zhang@epfl.ch; Nicolas Roussel, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, nicolas.roussel@epfl.ch; Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, wenzel.jakob@epfl.ch.

0730-0301/2023/12-ART212 $15.00
https://doi.org/10.1145/3618385

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

Additional Key Words and Phrases: differentiable rendering, geometry reconstruction

ACM Reference Format:
Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Trans. Graph.* 42, 6, Article 212 (December 2023), 14 pages. https://doi.org/10.1145/3618385

1 INTRODUCTION

Differentiation is a core component of recent work on inverse rendering. Methods in this area combine the derivative of a rendering algorithm with a variant of gradient descent to optimize a tentative scene until renderings become consistent with observed data.

However, a frustrating problem that invariably arises when differentiating a Monte Carlo renderer is that the computed gradient is wrong, to the extent that even basic optimizations fail. More precisely, derivatives with respect to certain kinds of parameters including vertex positions, curve control points, and transformation matrices are contaminated by bias. The relevant shared characteristic of these examples is their effect on *silhouettes*, which refers to the outlines of objects that occlude the background, other objects, or even a part of themselves. At silhouettes, small perturbations to a ray's origin or direction cause a sudden jump in the closest ray-surface intersection.

The combination of this property with the ubiquitous Monte Carlo method leads to a curious mathematical artifact: changing "problematic" parameters causes sudden jumps in sampled ray intersections, which should have a noticeable influence on the rendered

image and its derivative. However, the effect is lost under differentiation, since the change at individual Monte Carlo samples is discontinuous. This defect is not present in the original continuous theory, which motivates techniques that seek to remove the discrepancy.

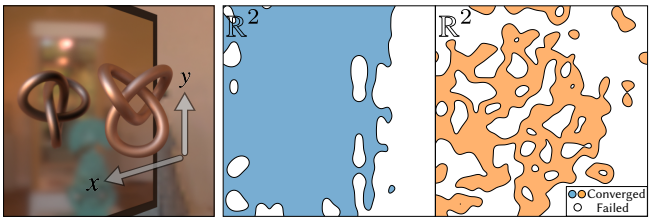
Prior work in this area falls into two broad categories: *boundary-based methods* generate additional Monte Carlo samples on silhouettes to account for their effect, while *area-based methods* compute a modified integral using ordinary samples covering the full domain.

Our method is a *hybrid* in this classification: it ingests ordinary samples and converts them into boundary samples. We feed it with the output of standard ("*primal*") rendering strategies, which follows Griewank and Walther [2008]'s rule #2 for automatic differentiation: "*What is good for function values is good for their derivatives.*". In particular, BRDF-, emitter-, and multiple importance sampling [Veach and Guibas 1995] are indispensable tools for efficient primal rendering, and we similarly wish to use them to improve the quality of gradient estimates.

Our method builds on the path-space differentiable rendering by Zhang et al. [2020] and makes both theoretical and practical contributions to this framework. We revisit the local parameterization of the boundary integral and bring the primary equation into its ultimate reduced form. We also examine visibility discontinuities arising from the interior of smooth geometry and propose a generalized local formulation that subsumes all cases.

Contemporary rendering systems support a great variety of shape representations. Our technique requires users to supply a projection operator per type that maps ray segments onto a nearby silhouette. We demonstrate how this modular approach enables differentiable rendering of triangle meshes, implicit functions, and cylindrical fibers based on Bézier curves, while producing gradients with an exceptionally low level of variance compared to prior work. Besides improving computational efficiency, low gradient variance can also improve the stability of optimizations (Figure 2).

Following a review of prior work, Section 3 will first introduce the core idea in a simple 2D setting followed by a mathematical formalization of the boundary integral in Section 4. Sections 5 covers algorithmic and implementation-level details, and Section 6 provides



Optimization task | With high-quality gradients | With low-quality gradients

Fig. 2. **Gradient quality and robustness.** Stochastic optimizers like Adam support noisy gradients, but excessive variance can be detrimental. In this example, we infer the X/Y offset of the knot shape from a single reference view. The middle and right images depict the convergence status following initialization at the associated point within the parameter domain (white indicates a failure to converge to the known parameter value). The only difference between these experiments is the quality of the gradient, which was computed using either finite differences with 128 samples/pixel (middle), or a reparameterization with 32 auxiliary rays [Bangaru et al. 2020] (right).

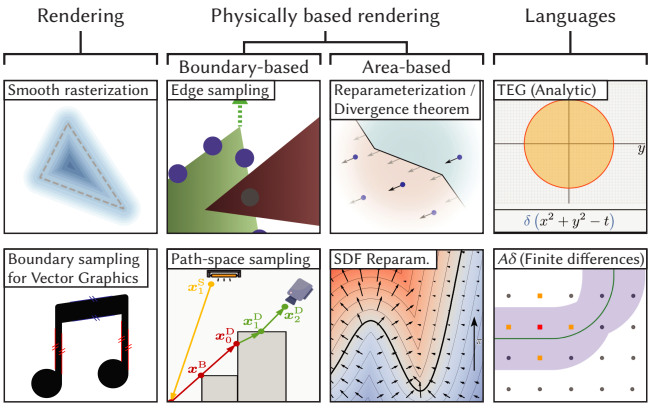


Fig. 3. **A taxonomy of prior work.** Visible discontinuities can be handled by smoothing boundaries [Liu et al. 2019] or explicitly sampling them [Li et al. 2020]. The physically-based setting features indirectly observed discontinuities. Previous boundary-based sampling methods used 6D Hough trees [Li et al. 2018] or path space [Zhang et al. 2020]. Alternatively, an area-based integral can be reparameterized to freeze discontinuities [Loubet et al. 2019], which is equivalent to an application of the divergence theorem [Bangaru et al. 2020]. SDFs are particularly amenable to reparameterization [Vicini et al. 2022; Bangaru et al. 2022]. Recent differentiable programming languages differentiate discontinuous integrals using analytic methods [Bangaru et al. 2021] and finite differences [Yang et al. 2022].

experimental comparisons to prior area and boundary-based methods. An open-source implementation of this method is available at `https://github.com/mitsuba-renderer/mitsuba3`.

This article includes numerous visualizations of *forward derivatives* (for instance, see Figure 1). These characterize the change of rendered pixels in response to a perturbation of a single scene parameter. In reality, the proposed method would more likely be used to compute *backward derivatives* of all parameters relative to a given image loss. We opt to show forward derivatives since they more intuitively convey information about sources of bias and variance.

2 RELATED WORK

This section reviews relevant prior work for handling discontinuities in derivatives of rendering algorithms. Figure 3 organizes a subset of them in a visual taxonomy.

2.1 Rendering

Differentiable rasterization. In its simplest form, rasterization converts polygonal meshes into fragments that are shaded and then depth-tested to form a raster image. Differentiable rasterization [Loper and Black 2014; Kato et al. 2018; Liu et al. 2019; Cole et al. 2021; Petersen et al. 2022] replaces specific steps of this process with smooth analogs, e.g. by blurring polygon boundaries or smoothly blending fragments. Differentiable rasterization can be made impressively efficient [Laine et al. 2020] but does not account for indirect effects like shadows or indirect illumination.

Differentiable vector graphics. Li et al. [2020] propose a differentiable antialiasing technique for vector graphics built from cubic Bézier curves. They apply the Reynolds transport theorem [1903] to

turn derivative contributions from boundaries into a 1D boundary integral and sample it using the Monte Carlo method.

Other methods for primary visibility. A large number of prior works propose specialized methods for differentiable rendering of SDFs, volumes, neural fields, and other representations involving mask or silhouette losses to account for visible discontinuities. To keep the discussion focused, we skip them along with work on inverse rendering pipelines, losses, and regularization techniques.

2.2 Physically based rendering

The physically based setting introduces five additional challenges:

1. Methods must account for *indirectly observed discontinuities* on a higher-dimensional space of light paths. For example, the amount of shadowing below an object depends on the degree to which its geometry occludes surrounding sources of direct and indirect illumination. Figure 4 visually decomposes the components of the resulting derivative.
2. The reflectance integral at every scene position observes a different set of silhouettes. This means that detecting and storing these silhouettes ahead of time is not straightforward.
3. Compared to the vector graphics setting [Li et al. 2020], there are too many discontinuities to enumerate. They must be handled randomly, ideally in some way that guides the computation towards important ones.
4. Not all discontinuities are equal: their contribution is proportional to the discontinuous change in the primal integrand (Figure 5b), which is often highly heterogeneous.
5. As the scene complexity grows, an increasing fraction of *potential silhouettes* becomes irrelevant because they are themselves occluded by surrounding geometry.

No existing solution fully addresses this challenging set of constraints. The following ideas have been proposed in the past:

Boundary sampling. Li et al. [2018] employ a 6D Hough tree to sample potentially observed boundaries relative to a given scene position. This pioneering method was the first to address indirect visibility, though it comes with various practical limitations: poor scaling of the high-dimensional data structure with respect to scene complexity and the difficulty of drawing samples proportional to their contribution while excluding invisible silhouettes.

Zhang et al. [2020] subsequently approached the problem through Veach's [1997] path space formalism which reveals that silhouette paths are more easily generated "from the middle" by sampling a direction tangential to any edge and expanding it outwards to form a complete path connecting the sensor to an emitter. The core sampling problem entails selecting an edge position t and a tangential direction θ, ϕ proportional to the expected value of a Monte Carlo estimator $\langle f(t, \theta, \phi) \rangle$. This is not easy, as $E[f]$ can fluctuate by many orders of magnitude. Zhang et al. [2020] tabulate $E[f]$ on a dense 3D grid to sample relevant parts of this domain.

The expense of tabulating at a sufficiently fine resolution can become a critical bottleneck, which has motivated subsequent work on adaptive representations that can more easily accommodate narrow peaks in $E[f]$. Yan et al. [2022] generate a sequence of separate

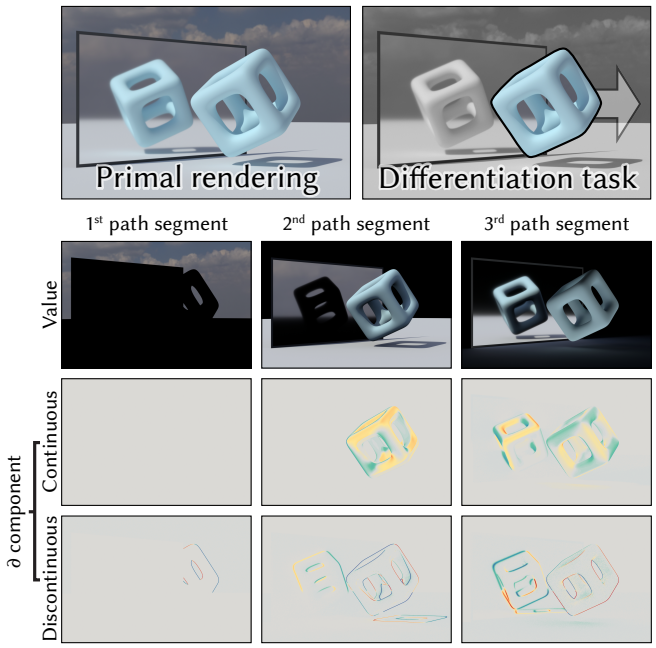


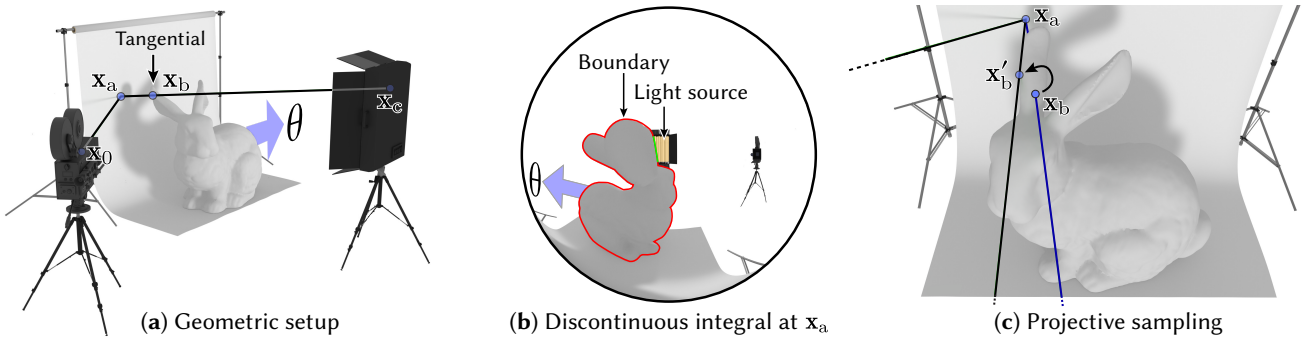
Fig. 4. **Image and derivative decomposition.** A physically based renderer must account for various illumination sources, which further grows when considering their derivative. The columns below separate the contribution of visible emitters (**left**) and shapes subject to direct (**middle**) and indirect illumination (**right**). The continuous derivative (**row 3**) is easily handled using automatic differentiation or adjoint methods [Nimier-David et al. 2020; Vicini et al. 2021]. The contribution of discontinuities (**bottom row**) poses severe challenges and is the primary focus of this article.

kd-trees over chains of adjacent edges and subdivide them recursively until random samples in leaf nodes are sufficiently uniform. As with any other adaptive integration technique, the method may fail to detect sharp peaks and stop the subdivision prematurely.

Current methods in this category only support polygonal meshes. This is unfortunate, since continuous shape representations offer desirable qualities: implicit representations like *signed distance functions* (SDFs) support topological changes and exhibit improved stability in non-regularized optimizations compared to discrete meshes [Vicini et al. 2022]. Despite these benefits, it is still unclear how the idea of boundary sampling could be adapted to continuous representations that lack a notion of discrete edges.

Reparameterization. Instead of sampling the boundary, methods in this category reparameterize the interior integral using coordinates that smoothly deform the evolving domain in such a way that discontinuous boundaries become stationary. Boundary-related derivatives then arise from the interior deformation, which is captured by the parameterization's Jacobian determinant. The first work in this sequence [Loubet et al. 2019] proposed a heuristic parameterization, which was later improved using a construction with lower bias [Bangaru et al. 2020].

Visibility changes at object silhouettes are not the only source of discontinuities: for example, geometric normals can introduce shading discontinuities at the edges of polygonal meshes, and this can



(**a**) Geometric setup (**b**) Discontinuous integral at $\mathbf{x}_a$ (**c**) Projective sampling

Fig. 5. **High-level overview. (a)** Rendering of a bunny with translation parameter θ. Increasing the value of θ brightens the partially shadowed surface position $\mathbf{x}_a$. Image (**b**) shows the spherical integral that determines the reflectance at $\mathbf{x}_a$, which shows how increasing θ shifts the silhouettes (red curves) towards the left and reveals more of the partially blocked light source. To account for this effect during differentiation, one can place additional Monte Carlo samples directly onto the boundary by generating tangential path segments $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$. Our method leverages standard primal sampling techniques to find relevant parts of this boundary. The example in (**c**) shows a sample from a direct illumination strategy (blue) that was ultimately unsuccessful due to occlusion. Our method takes this segment and projects it onto a nearby silhouette.

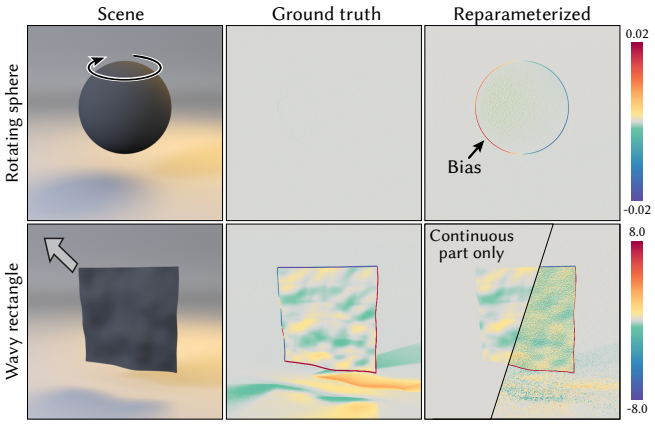


Fig. 6. Reparameterizations [Bangaru et al. 2020; Loubet et al. 2019] inject variance and bias into the derivative computation. **(top.)** This uniform rotating sphere should not produce visible derivatives. The reparameterization fails to identify this and produces biased output. **(bot.)** The derivative of a translating wavy rectangle has interior and boundary components. A naïve Monte Carlo estimator misses the boundary derivative but produces an otherwise well-converged estimate of the interior part. The reparameterized version fixes the boundary at the cost of a global variance increase.

similarly be handled by a reparameterization of the interior known as the *material form* [Zhang et al. 2020]. We only focus on visibility-induced discontinuities but note that these ideas are orthogonal and could be combined to handle both types of discontinuities.

Reparameterization-based methods have clear benefits: they work with any geometric representation and are tightly integrated into the original simulation, which allows them to only focus on visible scene geometry. On the flipside, they must trace many auxiliary rays to sense the motion of surrounding geometry, which is necessary to create appropriate coordinate maps for a given path segment. The numerical aspects of this process are somewhat arcane and involve convolutions with singular kernels that inject a substantial amount of variance into the differentiable rendering process. This variance even impacts the interior parts far from silhouettes. We have found it difficult to configure these methods to strike an acceptable balance between variance and bias (see Figure 6). In some instances, reparameterizations can be tailored to the underlying geometric representation, and two recent works explore the specialization to SDFs [Vicini et al. 2022; Bangaru et al. 2022].

Analytic visibility. A third way of addressing the issue involves substituting the Monte Carlo method with analytic solutions that inherently yield correct derivative when differentiated [Zhou et al. 2021]. The requirement of closed-form integrability is relatively stringent and restricts this approach to directly illuminated surfaces with simple material models.

Differentiable programming. Other works have explored the effect of discontinuous decision boundaries (e.g., an `if`) in the context of shaders [Yang et al. 2022] and domain-specific languages with an integration primitive [Bangaru et al. 2021].

We believe that some ideas in this article may apply to this more general setting, e.g., by repeatedly running a program with perturbed inputs and adding a root-finding iteration that projects samples onto the decision boundary to account for its derivative.

3 OVERVIEW

A brief comment on terminology: our method computes a *boundary integral* to account for the effect of visibility during differentiation, but this integral itself has an interior and a boundary term. This is because both the interior (a 2D domain) and boundary (a 1D curve) of a surface (e.g., a curved patch) can impact visibility. To reduce severe terminological overload of the word *boundary*, we will refer to the 1D domain of the boundary integral as the *perimeter*.

We now turn to an example to review the mathematical nature of the problem and present the high-level idea of our approach. Figure 5a illustrates the geometric setup: a camera observes an object casting a smooth shadow due to a nearby area emitter. A solitary scene parameter θ controls the shape's horizontal translation[1].

[1]This single-parameter example is only for illustration. Normal usage of a differentiable renderer involves differentiation with respect to millions of parameters.

The following hemispherical cosine-weighted integral determines the diffuse outgoing radiance $L_o(\mathbf{x}_a)$ at position $\mathbf{x}_a$:

$$L_o(\mathbf{x}_a) = \rho \int_{\mathcal{H}^2} L_i(\mathbf{x}_a, \omega)\, \mathrm{d}\boldsymbol{\omega}^\perp, \tag{1}$$

where L_i denotes the incident radiance that implicitly depends on θ, and ρ refers to the object's diffuse albedo. Increasing θ moves the shadow away from $\mathbf{x}_a$, which in turn increases the value of $L_o(\mathbf{x}_a)$. The derivative $\partial_\theta L_o$ expresses the precise rate of change and is given by an integral of a derivative (a.k.a. "differentiation under the integral sign") and an additional boundary term illustrated in Figure 5b:

$$\partial_\theta L_o(\mathbf{x}_a) = \rho \int_{\mathcal{H}^2} \partial_\theta L_i(\mathbf{x}_a, \boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}^\perp + \rho \int_{\mathcal{B}} (\partial_\theta \mathbf{x}(t) \cdot \mathbf{n}(t))\, \Delta L_i\, \mathrm{d}t. \tag{2}$$

The first term is readily computable using existing methods like automatic differentiation or Path Replay Backpropagation [Vicini et al. 2021], but it only plays a minor role in this example. The dominant second term integrates the discontinuous change in incident radiance ΔL_i across the evolving boundary $\mathcal{B}$ scaled by its perpendicular velocity. The computation of this boundary term presents serious difficulties, as detailed in Section 2.

Our work on this topic was prompted by the realization that a boundary projection operation could greatly reduce these difficulties. Such a projection would instantly turn any existing technique for the interior into a specialized method for boundaries that could then be used to compute the troublesome integral.

Figure 5c depicts a possible application of this idea: suppose that a light source sample turns out to be occluded as seem from position $\mathbf{x}_a$. This provides a helpful clue for the differential phase: if the occlusion is only partial, we may be able to "walk" towards the associated boundary to account for its derivative contribution.

While such a projection can surely be constructed, this immediately raises another question: how would one actually use it in the Monte Carlo context? The estimator would need to consider the probability $p(t)$ of the generated boundary sample $t \in \mathcal{B}$ per unit length (or more accurately, its *reciprocal*). The density $p(t)$ represents an intricate marginal dependent on both the input 2D density and the specifics of the projection. We cannot expect that this marginal will be available in closed form.

We experimented with unbiased methods that draw auxiliary samples to estimate the reciprocal density [Booth 2007] but found the overheads of this process to be unacceptably high. In essence, the intricacy of the projection is such that attempts to characterize it precisely negate the benefit of having a projection to begin with.

What, then, if we tried to model $p(t)$ *imprecisely*? This idea is the basis of our method, which consists of three steps: the first is an ordinary primal rendering step that additionally projects samples onto nearby boundaries (i.e., it draws samples from $p(t)$). The second builds a closed-form *guiding distribution* $h(t) \approx p(t)$ from the projections. The last step computes the boundary integral by importance sampling the guiding distribution. If $h(t)$ covers relevant parts of the domain, such a method will still produce unbiased estimates despite the approximate nature of h. Section 5 provides algorithmic and implementation-level details needed to turn this high-level idea into a practical method.

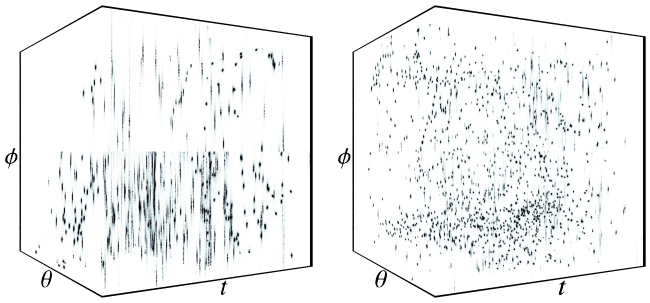


Fig. 7. **Boundary sample space.** Visualization of the boundary integrand of the Bunny and Filigree meshes from Figure 10 on *boundary sample space*. This 3D domain is parameterized by edge position $t \in \partial\mathcal{A}$ and a spherical direction $(\theta, \phi) \in \mathcal{S}^2$. Note the extremely sparse and high-frequency nature of this function (white corresponds to an integrated value of 0 along a ray viewing the 3D volume). The supplemental video contains an animated version of this figure.

3.1 Discussion

This high-level summary already provides enough context to discuss the method's conceptual advantages and disadvantages.

1. **Modularity.** Our method does not rely on any particular representation and fits into the modular architecture of contemporary rendering systems. To support a new geometric primitive, the user must implement two, rarely three operations: the previously mentioned projection, and a parameterization of the geometric perimeter (if present) or curved interior (if present).

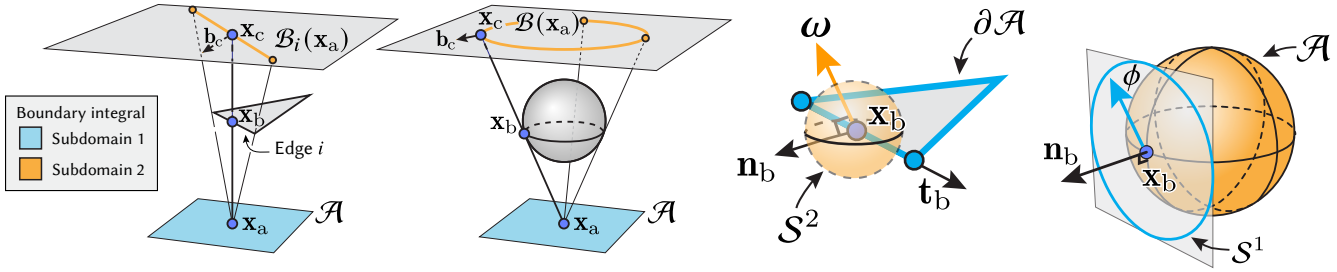
2. **Heterogeneity**. The value of the boundary integrand tends to fluctuate by many orders of magnitude. Figure 5b illustrates this: the boundary curve has a considerable arclength, but only the small segment directly adjacent to the light source contributes.

   Our method can leverage an existing ecosystem of sampling techniques designed to cover high-valued regions of the primal integrand with high probability, which in turn benefits the boundary integral. Without added effort on our part, the computation stays confined to visible scene geometry, which has been a notable challenge in previous work.

3. **Guiding.** Despite the advantage mentioned above, we find that the density p resulting from the projection is often suboptimal and *not sufficiently proportional* to the value of the actual boundary integrand. The decoupled nature of the guiding distribution $h \neq p$ provides the means to address this problem, since it allows us to adjust h retroactively using more accurate knowledge about the boundary integrand.

   Figure 7 visualizes the exceedingly sparse nature of the integrand on the *boundary sample space* covered by our guiding scheme. The primary role of the projection is to find the narrow nonzero regions. We subsequently evaluate the actual integrand at the projected locations to create the final guiding distribution.

4. **Sampling efficiency.** Our method improves statistical efficiency compared to prior path-space methods [Zhang et al. 2020; Yan et al. 2022], and it improves handling of the interior component compared to reparameterizations [Loubet et al. 2019;



(**a**) Non-local (perimeter) (**b**) Non-local (interior) (**c**) Local (perimeter) (**d**) Local (interior)

Fig. 8. **Formulations and terms of the boundary integral**. Visibility-related derivatives arise from the perimeter (e.g., discrete edges of a triangle mesh) and the interior of shapes (e.g. the surface of an ellipsoid). Path-space methods compute an integral over tangential path segments to account for them. Decomposing the integration domain (blue and orange sets) reveals different formulations: **(a)** For the perimeter component, one can integrate over source points $\mathbf{x}_a \in \mathcal{A}$ and the "shadow" $\mathbf{x}_c \in \mathcal{B}_i(\mathbf{x}_a)$ cast by a discrete edge i. (**b**) This also generalizes to the interior, but parameterizing and sampling the projected boundary $\mathcal{B}(\mathbf{x}_a)$ is difficult in general. (**c**) The local formulation instead evaluates a spherical integral at boundaries $\mathbf{x}_b \in \partial\mathcal{A}$ without explicit consideration of the neighboring vertices $\mathbf{x}_a$ and $\mathbf{x}_c$. (**d**) The interior can be handled analogously but requires a different partition into an integral over surface positions (orange) and tangential directions (blue). We propose a new local boundary integral that accounts for this component.

Bangaru et al. 2020]. The latter methods must reparameterize every ray at a significant cost to trace auxiliary rays, and this furthermore injects variance affecting the interior derivatives (Figure 6). Our method can skip the projection in regions that are far away from the boundary, and it preserves the statistical quality of interior derivatives.

Our method's main drawbacks all stem from the central projection step: conversion of interior to boundary samples only works well when there is sufficient surface area to collect samples, and when the projection itself is well-behaved. For example, a thin shape like a blade of grass has a large boundary arclength to surface area ratio and may not receive a sufficient number of samples. Similarly, a function that maps every interior point onto a fixed silhouette location is technically a projection, but not one that is conductive to solving our problem. The first point is a limitation of our method. We address the second point by presenting high-quality projections for diverse geometric representations in Section 5.

Before delving into the details of projection and guiding, we must first turn to a theoretical aspect of the problem that will enable generalization to a wider array of shapes.

4 A LOCAL BOUNDARY INTEGRAL

Zhang et al.'s [2020] path space formulation of differentiable rendering decouples the effect of boundaries from their interior, enabling targeted computation of each part using specialized methods. Their equation (43) provides the starting point of our method.

This equation relates the change in pixel intensity I due to visibility changes with respect to a perturbation of a scene parameter θ. Expressed with respect to the path segment $(\mathbf{x}_a, \mathbf{x}_c)$, it states

$$\frac{\partial I}{\partial \theta} = \int_{\mathcal{A}} \int_{\mathcal{B}(\mathbf{x}_a)} L_i(\mathbf{x}_a, \mathbf{x}_c)\, G(\mathbf{x}_a, \mathbf{x}_c) W_i(\mathbf{x}_c, \mathbf{x}_a) \, (\partial_\theta \mathbf{x}_c \cdot \mathbf{n}_c)\, \mathrm{d}l(\mathbf{x}_c)\, \mathrm{d}A(\mathbf{x}_a). \quad (3)$$

The integral is over segments $(\mathbf{x}_a, \mathbf{x}_c)$ that make contact with surface boundary along the way. The domain $\mathcal{B}(\mathbf{x}_a)$ describes the "shadow" of this boundary and is further composed of $\mathcal{B}(\mathbf{x}_a) = \cup_i \mathcal{B}_i(\mathbf{x}_a)$ (one set $\mathcal{B}_i$ per edge) when the scene consists of discrete geometry (Figure 8a). The functions L_i and W_i refer to the incident radiance and importance and can be computed using existing primal estimators, G is the standard geometric term [Veach 1997], and the inner product measures the perpendicular speed of the "shadow" at $\mathbf{x}_c$ (see Figure 8a).

In practice, it is simpler and far more efficient to build boundary segments "outwards" starting from the tangential point $\mathbf{x}_b$, and Zhang et al. therefore also propose a *local* formulation of the perimeter contribution (Figure 8c).

However, their derivation [Zhang et al. 2020, Appendix 1] is incomplete in the sense that the final equation contains derivatives arising from a ray tracing operation performed using autodiff. The presence of this complex step obscures simplification opportunities that can bring this equation into its ultimate reduced form. These simplifications can also have implications on sampling efficiency.

Our contribution here is two-fold: we re-derive the local formulation of the perimeter to reveal its inherent simplicity, and we propose the first local formulation of the interior. Combining both sources of derivatives leads to the following complete expression, which provides the theoretical basis of our method.

$$\frac{\partial I}{\partial \theta} = \int_{\partial\mathcal{A}} \int_{\mathcal{S}^2} L_d(\mathbf{x}_b, \boldsymbol{\omega})\, W_i(\mathbf{x}_b, \boldsymbol{\omega})\, \sin\phi\, (\partial_\theta \mathbf{x}_b \cdot \mathbf{n}_b)\, \mathrm{d}\boldsymbol{\omega}\, \mathrm{d}l(\mathbf{x}_b) + \int_{\mathcal{A}} \int_{\mathcal{S}^1} L_d(\mathbf{x}_b, \phi)\, W_i(\mathbf{x}_b, \phi)\, \kappa(\phi)\, (\partial_\theta \mathbf{x}_b \cdot \mathbf{n}_b)\, \mathrm{d}\phi\, \mathrm{d}A(\mathbf{x}_b). \quad (4)$$

The derivation of this expression is somewhat technical, and we refer the reader to the supplemental material for a detailed proof.

The term L_d stands for the radiance difference between foreground and background $L_d(\mathbf{x}_b, \omega) = L_o(\mathbf{x}_b, \omega) - L_i(\mathbf{x}_b, -\omega)$. In the perimeter term (top integral), $\sin\phi$ refers to the angle between $\boldsymbol{\omega}$ and the boundary tangent $\mathbf{t}_b$ (Figure 8c). In the interior term, the angle $\phi \in \mathcal{S}^1$ parameterizes all relevant quantities over tangential directions at the surface position $\mathbf{x}_b$ (Figure 8d), and $\kappa(\phi)$ denotes the normal curvature. The following aspects are noteworthy:

1. Neither term involves the complex non-local domain $\mathcal{B}(\mathbf{x}_a)$.

2. Contrasting with Equation 3, the geometric term G is absent in both integrals, which means that variance arising from this factor[2] can be avoided in Monte Carlo methods. This is not obvious when looking at Equation (3), and the reference implementation of Zhang et al. [2020], e.g., tabulates G within its guiding distribution despite it canceling in subsequent steps.
3. When the scene geometry is smooth and closed, $\partial\mathcal{A} = \emptyset$ removes the first term. Polygonal meshes do not have curved interior ($\kappa = 0$) and hence do not require the second term.
4. The sine (perimeter) and curvature (interior) terms resemble the cosine factor in the rendering equation: The influence of an edge with tangent $\mathbf{t}_b$ exerted in direction $\boldsymbol{\omega}$ tends to zero as $\mathbf{t}_b \to \boldsymbol{\omega}$, since the edge becomes invisible. Likewise, the curvature term accounts for foreshortening in the mapping between silhouette positions and scene positions observing this silhouette.

5 METHOD

With this background in place, we can now delve into the details of interior and guided sampling. We discuss the projection operation last, since it requires specialization to various geometric representations. This section already presents a number of results that examine the quality of computed gradients to motivate algorithmic design decisions. These are followed by more comprehensive end-to-end optimization results in Section 6.

Input sampling strategies. Our method repurposes interior sampling strategies into specialized strategies targeting the boundary. But which distributions should be used as the input of this process? In the setting of primal rendering, it is common to rely on combinations of multiple strategies like BSDF and emitter sampling via *multiple importance sampling* (MIS) [Veach 1997]. Figure 9 explores this possibility, indicating that the established intuition of such combinations transfers to the differential setting.

Embracing imperfection. Section 3 introduced the concept of a path segment projection that maps an input segment $\mathbf{x}_a \to \mathbf{x}_b$ onto a nearby segment $\mathbf{x}_a \to \mathbf{x}'_b$ such that $\mathbf{x}'_b$ is located on a silhouette observed by $\mathbf{x}_a$. It is important to note that the current requirements for this operation are stricter than necessary, and that there could be benefits to relaxing them.

To see why, remember that we abandoned the idea of solving the boundary integral with respect to a specific position $\mathbf{x}_a$, since the density of projected samples was too expensive to characterize. The projections merely guide a subsequent integration phase that accounts for all surface positions $\mathbf{x}_a$ simultaneously. Thus, it suffices to find an approximate boundary segment $\mathbf{x}'_a \to \mathbf{x}'_b$ close to $\mathbf{x}_a \to \mathbf{x}_b$ (i.e., allowing for a slight shift of the origin $\mathbf{x}_a$ as well). This added flexibility can be used in two ways: First, the projection might fail. In such cases, we snap $\mathbf{x}'_b$ to the closest local boundary (e.g. a triangle edge) and select the tangential direction closest to the original direction of $\mathbf{x}_a \to \mathbf{x}_b$. Further, if a projection requires root-finding, we terminate the iteration after a few steps without waiting for convergence to machine precision. The former step reduces variance, while the latter reduces the cost of creating the guiding distribution.

[2]To see why, consider a rod with 45-degree inclination casting a shadow from an overhead directional light onto a ground plane. Prior work placed more samples on the far end of the rod's silhouette, which is undesirable as the contribution per unit arclength is uniform.

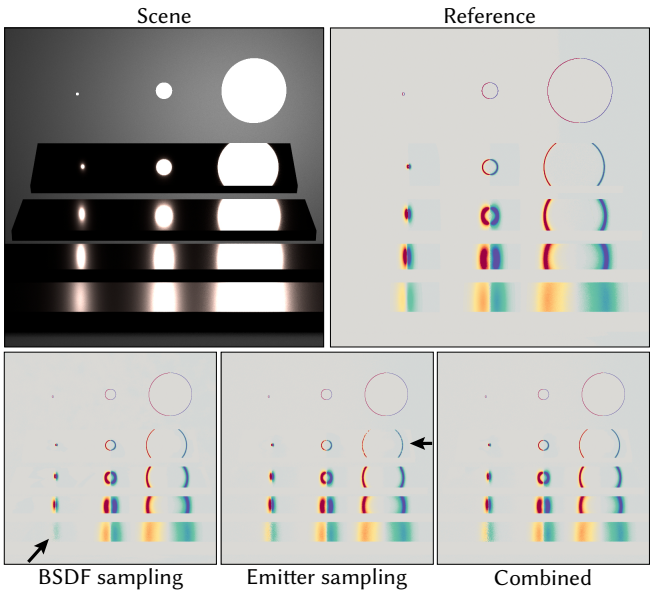


Fig. 9. **Impact of the interior sampling distribution**. This experiment analyzes the derivative of a classic test scene by Veach with respect to a horizontal translation. Closer investigation reveals familiar failure modes: the projected BSDF sampling strategy fails to generate a sufficient number of silhouette samples to cover the rough reflection of the smallest sphere, while emitter sampling presents the opposite issue in the smooth reflection of the largest sphere. Analogous to multiple importance sampling [Veach 1997] for primal rendering, it can be beneficial to project a mixture of both.

5.1 Guiding distribution

Given a set of projected samples, the next major step consists of condensing this information into a representation that supports efficient sampling and density evaluation.

5.1.1 Grid-based guiding. The simplest option is a 3D density grid, which can be a good choice when the integrand exhibits sufficient smoothness. Zhang et al.'s [2020] *path-space differentiable rendering* (PSDR) method likewise relies on a grid.

One notable advantage of grids is that the projection can directly accumulate projections without having to store the samples themselves, which can enable high-quality statistics accumulation in memory-constrained environments. Conversely, a high resolution 3D grid can be very memory-intensive and tends to become a bottleneck in complex scenes requiring a fine discretization to resolve sparse features.

Figure 10 presents a first set of results for grid-based guiding. Each pair of rows shows the gradient with respect to a horizontal translation in the first row, followed by error visualizations comparing to a finite difference reference in the second row. We use two different color scales throughout the paper: a standard red/gray/blue scale for the error images in the lower rows (with gray representing zero error), and a more varied color scale for the gradient images due

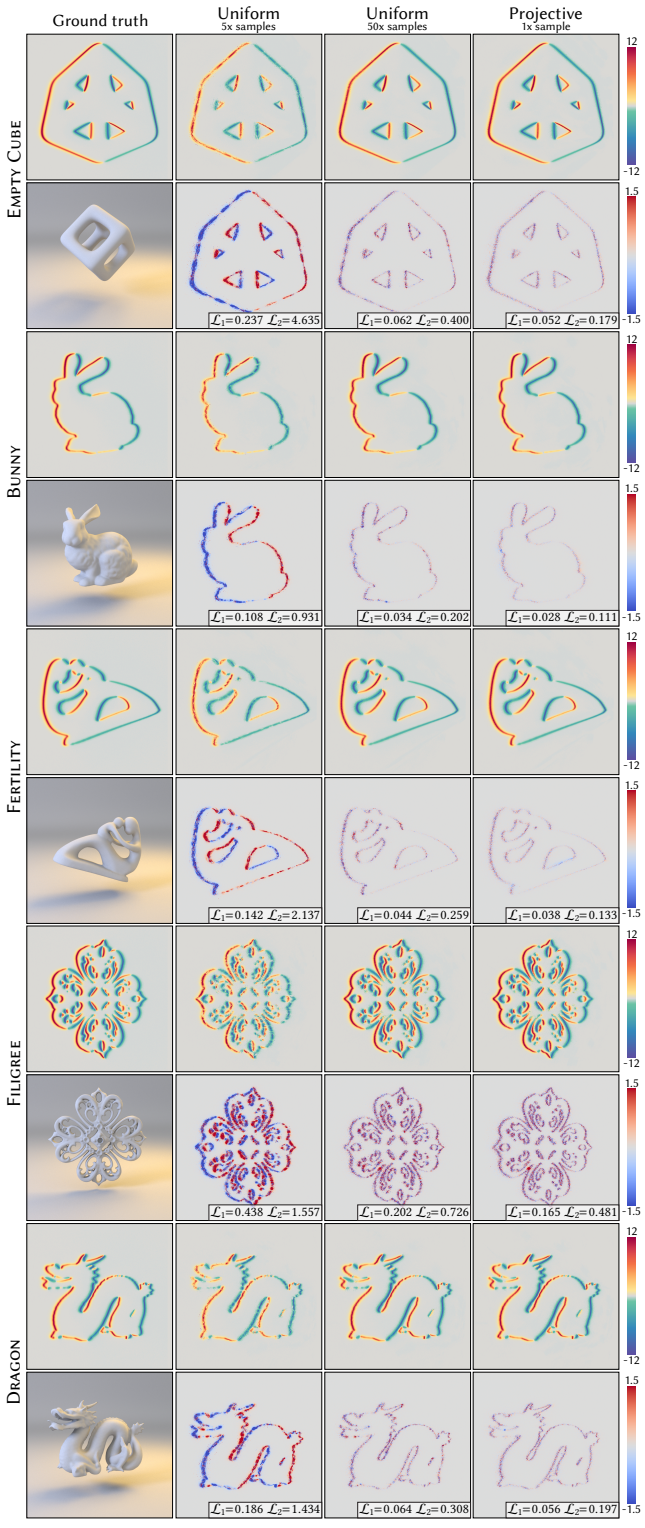


Fig. 10. Qualitative and numerical evaluation of grid-based guiding comparing uniform [Zhang et al. 2020] and projective sampling on meshes of varying geometric complexity. Each pair of rows shows image gradients followed by a visualization of the associated error. See Table 1 for timings.

to their large dynamic range. The $\mathcal{L}_1$ and $\mathcal{L}_2$ annotations denote the gradient MAE and RMSE values compared to the reference. The primal rendering is illustrative and uses different viewpoint and lighting (Section 6 provides more details on the test scene setup).

All results in Figure 10 use a fixed grid with $10000 \times 100 \times 100$ voxels for (t, θ, ϕ). The only difference is the number of accumulated samples, and whether or not projective sampling was used. The second column, labeled "*uniform, 5x samples*" shows the PSDR baseline[3] using five samples per voxel. The last column employs our projection for triangle meshes and was generated using less time than the competing methods (see Table 1 for precise timings). We also use a lower number of samples to populate the grid given the added cost of the projection. The middle result shows the quality improvement that could be obtained by PSDR when using many more samples to detect fine features within voxels, though this multiplies the computation time by a factor of nearly 10×. These results show that projective sampling consistently outperforms PSDR by a factor of 2.5-4.5× (MAE), which grows to a factor of up to 25× in RMSE due to the presence of outliers.

5.1.2 Hierarchical guiding. Figure 7 (on page 5) previously visualized the intricate structure of the integrand on *boundary sample space*. For polygonal meshes, this 3D space consists of a single parameter $t \in [0, 1]$, which parameterizes the set of mesh edges $\partial\mathcal{A}$, along with a direction (θ, ϕ) represented in spherical coordinates. Sparse features in this domain indicate complexities of the input scene, including:

1. BSDFs with narrow peaks.
2. Strongly peaked emitters including directional sources and environment maps featuring the sun.
3. Second-order visibility effects, where silhouettes are themselves occluded by surrounding geometry.
4. Discontinuities in the edge parameterization.

The last two points become more pronounced as the geometric complexity increases. For example, although a sphere is a very simple shape, its discretization into a polygonal mesh can generate an extremely challenging integrand.

Yan et al. [2022] propose two ways to mitigate these difficulties. First, they recommend rearranging the edges into longer connected chains, which reduces the number of discontinuities in the t parameter. We have not incorporated this optimization into our method, though it should be beneficial here as well.

Secondly, they replace the grid with a set of kd-trees (one tree per edge chain) to adapt to fine features of the 3D domain requiring increased resolution. The construction of this tree is top-down and resembles that of an adaptive quadrature rule: a recursive splitting criterion subdivides the current node until the integrand becomes sufficiently smooth. The failure mode of this approach is also that of adaptive quadrature: a stopping criterion based on local evaluations will sometimes fail to detect sharp peaks and stop the subdivision prematurely.

Our hierarchical guiding uses a set of 100 octrees, each covering an equal-sized interval of the first axis to handle heterogeneity

[3]We use our PSDR reimplementation, which outperforms the reference code. We also adapted it to use our improved local boundary integral formulation from Equation 4.

arising from the parameterization of edges, fibers, and the concatenation of multiple objects into a shared boundary sample space. The particular type of hierarchy is not a critical factor (we merely chose octrees since their construction is easy to parallelize on the GPU).

The main difference lies in *how this hierarchy is constructed*. Our hierarchy construction is bottom-up, starting from a set of projected samples concentrated at the sparse features in boundary sample space. We then simply subdivide each node until reaching a maximum depth (9), or until the node contains one or fewer samples, which yields a partition matching the spatio-directional structure of the integrand. In other words, unlike grid-based guiding, where a projection was used to accumulate density into the voxels of a 3D grid, we now employ the projection to cheaply construct a high-fidelity adaptive space partition. To estimate the actual value of the boundary integrand, we additionally draw 32 uniformly distributed samples within each leaf node, which does not require projections. In our experiments, this results in an average of $\sim 6.4 \times 10^7$ samples for a base budget of 10^8 projections, which roughly doubles the initial sample budget.

Figure 11 compares this scheme to the method of Yan et al. [2022] ("Adaptive quadrature") at equal time (conservatively, see Table 2 for precise timing values) and a grid-based baseline using projections. The experiment demonstrates that the octree significantly outperforms the grid-based baseline. The method of Yan et al. [2022] produces spatially non-uniform convergence with low error in some regions and significant outliers in others, causing a $\sim 190\times$ RMSE difference in the Neptune experiment.

5.2 Projection

We now discuss the operations needed to support a particular geometric representation:

1. A projection that maps a ray segment $\mathbf{x}_a \to \mathbf{x}_b$ onto a nearby segment $\mathbf{x}'_a \to \mathbf{x}'_b$ tangential at $\mathbf{x}'_b$.
2. A parameterization of the perimeter (if present), and a parameterization of the interior (for curved geometry).

5.2.1 Spheres. This is by far the simplest case, which can be helpful to test new implementations of projective sampling. Listing 1 provides pseudocode for the projection. Being smooth and closed, spheres only need a surface mapping (we use spherical coordinates).

5.2.2 Triangle meshes. We employ two different projection strategies for meshes: Jump and Walk.

The Jump projection is a Newton-style iteration based on a local linear approximation. We model the neighborhood of a surface position $\mathbf{p}$ using the parameterization $\tilde{\mathbf{p}}(u,v) = \mathbf{p} + u\partial_u\mathbf{p} + v\partial_v\mathbf{p}$ with an interpolated normal $\tilde{\mathbf{N}}(u,v) = \mathbf{N} + u\partial_u\mathbf{N} + v\partial_v\mathbf{N}$. Under the assumption of a fixed viewing direction, the silhouette of this approximation is a line in the parameter space, allowing us to analytically find a solution (u',v'). From the inferred silhouette point, we subsequently trace a perpendicular ray $(\tilde{\mathbf{p}}(u',v') + \varepsilon\tilde{\mathbf{N}}(u',v'), -\tilde{\mathbf{N}}(u',v'))$ to find the next intersection, at which point the procedure could be repeated.

The Walk projection is a greedy search strategy. We visit the three neighboring triangles and compute the angles between their normals and the viewing direction. Walk aims to gradually increase this

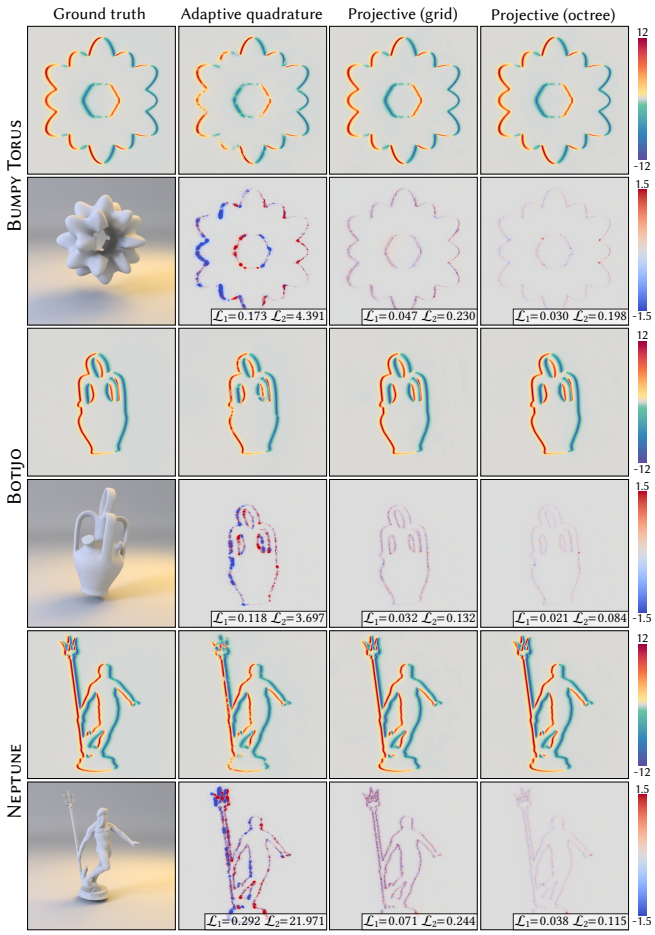


Fig. 11. Guiding data structure comparison including adaptive quadrature [Yan et al. 2022] (with edge sorting, MIS, and parameters set to use all available GPU memory), a uniform grid with projections, and an octree initialized with projected samples. See Table 2 for timings.

Algorithm 1. **Projection for spheres.** This code fragment demonstrates the interface of the projection operation for a simple geometric primitive.

```
class Sphere():
    def project(self, segment: Segment):
        O = segment.a # viewpoint
        P = segment.b # point on the sphere

        N = (P - self.center) / self.radius  # the normal at P
        CO = O - self.center # center to viewpoint vector
        CO_dir = (O - self.center) / norm(CO)
        proj_dir = normalize(N - dot(N, CO_dir) * CO_dir)

        # phi: the angle from C to the boundary relative to O
        sin_phi = r / norm(CO)
        B = CO_dir * sin_phi + proj_dir * sqrt(1 - sin_phi**2)

        return Segment(shape=self, a=O, b=B)
```

angle until reaching a perpendicular (90°) or back-facing triangle. However, consistently walking to the neighboring triangle with the largest angle tends to attract too many projections on certain mesh edges. Instead, we randomly pick between the two neighbors with the largest angle values, using them as discrete probabilities.

Both strategies only rely on local differential information. Jump is more aggressive and can quickly bypass plateaus and highly tesselated regions, while the smaller steps of Walk robustly detect nearby silhouettes even on bumpy geometry, where the extrapolated local model can be deceptive. Walk does not require ray tracing, which makes individual steps much faster compared to Jump.

In practice, we adopt a hybrid strategy: Walk for 30 steps and Jump once if no silhouette was found, followed by an additional 30 Walk steps. This strategy only requires a single ray tracing step and produces high-quality projections on all meshes presented in this paper. If the projection of the segment $\mathbf{x}_a \to \mathbf{x}_b$ fails to find a silhouette $\mathbf{x}'_b$, we keep the tentative position $\mathbf{x}'_b$ and generate a tangential segment $\mathbf{x}'_a \to \mathbf{x}'_b$ closest to the original direction of $\mathbf{x}_a \to \mathbf{x}_b$, as outlined at the beginning of Section 5. While more sophisticated projections could likely be pursued to handle various corner cases, we find that even this simple scheme already enables high-quality guiding.

5.3 Fiber curves

We model fibers using a parametric base curve $\mathbf{C}(v)$ and radius $r(v)$, which are both given by cubic B-spline interpolants (Figure 12). The cross-section of the surface for a fixed value of v yields a circle with center $\mathbf{C}(v)$, radius $r(v)$, and normal $\mathbf{C}'(v)$. Assigning an azimuth angle parameter u to this circle yields a C^1-continuous surface $\mathbf{M}(u, v)$. We ignore the curve endpoints (these can, e.g., be closed using spheres), in which case only the curved interior part of the local boundary integral in Equation (4) matters.

Projection. Given a viewpoint $\mathbf{O}$ and surface position $\mathbf{P} = \mathbf{M}(u_0, v_0)$, we fix v_0 and find the value of u that yields a silhouette projection, i.e., $\langle \mathbf{M}(u, v_0) - \mathbf{O}, \mathbf{n}(u, v_0) \rangle = 0$, where $\mathbf{n}$ is the parameterized surface normal. This equation can be expanded into the form

$$A \cos^2 u + B \cos u \sin u + C \cos u + D \sin u + E = 0,$$

where A, B, C, D, E are u-independent constants that can be computed analytically. The equation has an analytic solution when the fiber radius is constant. Otherwise, we run 20 bisection iterations, which is fast and robust.

5.4 Implicit functions

We also experimented with *signed distance functions* (SDFs) represented using a grid-based trilinear interpolant (Figure 12). This representation has two advantages:

1. **Speed.** Intersections can be computed with the help of a mesh proxy and analytic solutions within voxels, which avoids costly sphere tracing steps [Söderlund et al. 2022].
2. **Simplicity.** The low-order representation simplifies the mapping from $\mathcal{R}^2$ to the SDF surface.

The trilinear representation also has a clear disadvantage: geometric normals are discontinuous across voxel boundaries, which means that both interior and perimeter terms of Equation (4) must be considered[4]. Note that we do not use the *signed distance* property of the SDF representation, so much of the following could in principle generalize to other types of implicitly defined surfaces.

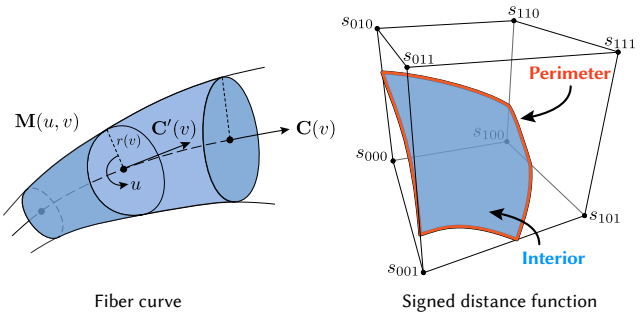


Fig. 12. **Smooth geometry**. Our new local formulation of the boundary derivative (Equation 4) enables differentiable rendering of smooth geometry, such as cylindrical fibers based on Bézier curves (left) and implicitly defined surfaces represented using a signed distance function (right). The latter case involves derivatives arising from the curved interior and potential normal discontinuities at voxel perimeters.

Parameterization. To define functions parameterizing the interior and perimeter, we first make the following observations:

1. **Interior.** Given the trilinear interpolation scheme, a given x, y coordinate within a voxel has at most one root along the z axis.
2. **Perimeter.** Similarly, the surface intersection with a voxel face can be uniquely parameterized by the face index and a perpendicular coordinate, which can be used to create a flattened 1D mapping across all possible perimeter curves reminiscent of the mesh case. Two additional dimensions represent the direction.

We use these properties to create a globally discontinuous per-voxel mapping. This mapping depends on the choice of a dimension used to parameterize the curve/surface, which is unstable when the chosen dimension is nearly perpendicular. We assign the most numerically stable dimension to each voxel based on the SDF gradient.

We did not realize a projection operation for SDFs but believe that such a step could be added along similar lines (related operations were also explored in prior work [Bremer and Hughes 1998]). Presented results for SDFs therefore use a grid data structure with uniform initialization.

6 RESULTS

We implemented our method on top of Mitsuba 3's [Jakob et al. 2022b] `cuda_ad_rgb` backend, using the underlying Dr.Jit [Jakob et al. 2022a] framework for forward/reverse-mode AD and GPU kernel compilation. All experiments ran on an AMD Ryzen 3970X workstation with an NVIDIA RTX 3090 graphics card.

Test scene. The test scene used to evaluate methods throughout the paper places the shape in front of a uniform area emitter. The camera views their reflection on a conductive plane with isotropic roughness (Trowbridge and Reitz [1975], $\alpha = 0.005$). This ensures

[4]In practice, such a low-order interpolation would be combined with a smooth shading normal approximation to suppress distracting seams at voxel boundaries. But this only fixes the smooth part of the differentiation problem–the visibility-related discontinuous part must consider the true geometric normal.

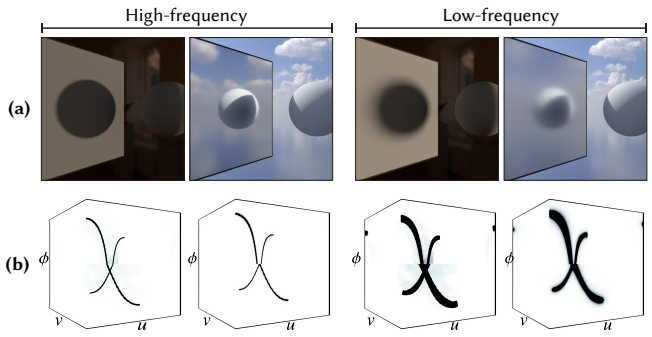


Fig. 13. **Shadows and reflections produce a qualitatively similar boundary integral. (a)** Primal rendering of scenes with shadows or reflections. **(b)** The corresponding boundary sample space. This figure demonstrates their similarity for both high-frequency (hard shadows, sharp reflections) and low-frequency (soft shadows, rough reflection) cases. Consequently, most of our gradient comparisons focus on the reflection case.

that the derivative image only contains contributions from indirectly observed discontinuities, which is the main focus of this work.

We prefer sharp reflections to investigate the performance of different methods: a soft material or shadow would blur the integrand in boundary sample space, potentially concealing inaccuracies or flaws in the method. As illustrated in Figure 13, shadows and reflections can produce a qualitatively similar boundary sample space. Instead of a reflective plane, one could therefore equivalently employ a diffuse receiver with a directionally peaked emitter.

Initialization time. Tables 1 and 2 list the time needed to initialize the guiding distributions for the previously discussed experiments in Figures 10 and 11. The runtime of the differential rendering phase is not noticeably impacted by the choice of guiding representation. In these gradient comparison experiments, derivative images were rendered using 256 samples per pixel (roughly ~ 0.8 second of rendering time), a number typically higher than necessary for inverse rendering tasks. We use this number to render ground truth quality derivative images as a validation of our method.

Fiber curve. Figure 14 showcases the influence of the guiding distribution for differentiable rendering of curves. It highlights the effectiveness of projective sampling, which outperforms uniform sampling even when the latter uses 50× more samples. We find that the octree representation can greatly reduce variance in complex cases like the Knitting Yarn experiment, where second-order visibility effects (occlusion of silhouettes) dominate.

End-to-end optimizations. Figure 15 demonstrates the use of our method as part of several complete reconstruction tasks. In the first two experiments, the optimization must infer the shape from a single view depicting the object and two shadows on a curved diffuse wall. We use the method of Nicolet et al. [2021] to smoothly evolve the triangle mesh and periodically re-mesh the geometry to a finer resolution to add more degrees of freedom. Listing 2 shows the high-level pseudocode of the optimization algorithm.

The final row of Figure 15 constructs a *Magic Lens* as in the work of Papas et al. [2012]. The objective of this task is to optimize the

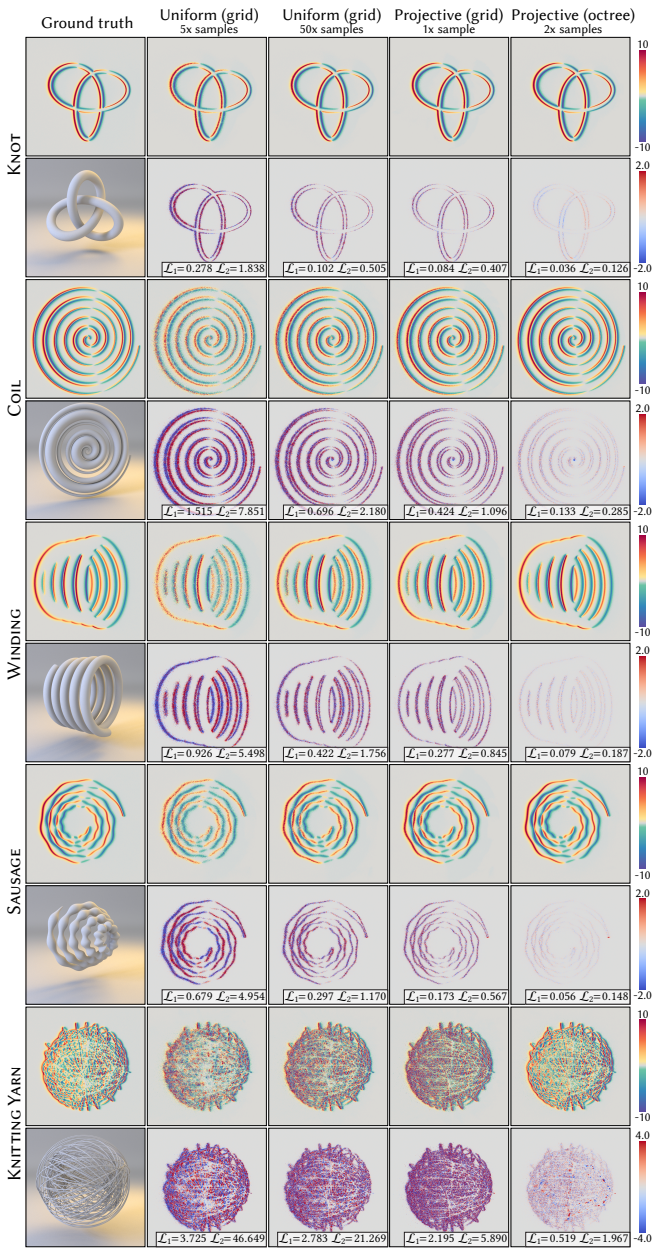


Fig. 14. Experimental validation of fiber derivatives analogous to previous experiments. See Table 3 for timings.

surface of a dielectric plate so that an object located behind the dielectric appears completely transformed into a different object when observed from a specific viewpoint. We place a hat behind the lens and optimize the lens so that the refraction resembles a bunny. In addition to lens' surface, we also optimize the hat's position. A sketch of this setup is shown in the leftmost column.

Influence of gradient quality. Figure 16 illustrates the influence of gradient quality in a challenging reconstruction task. The scene is

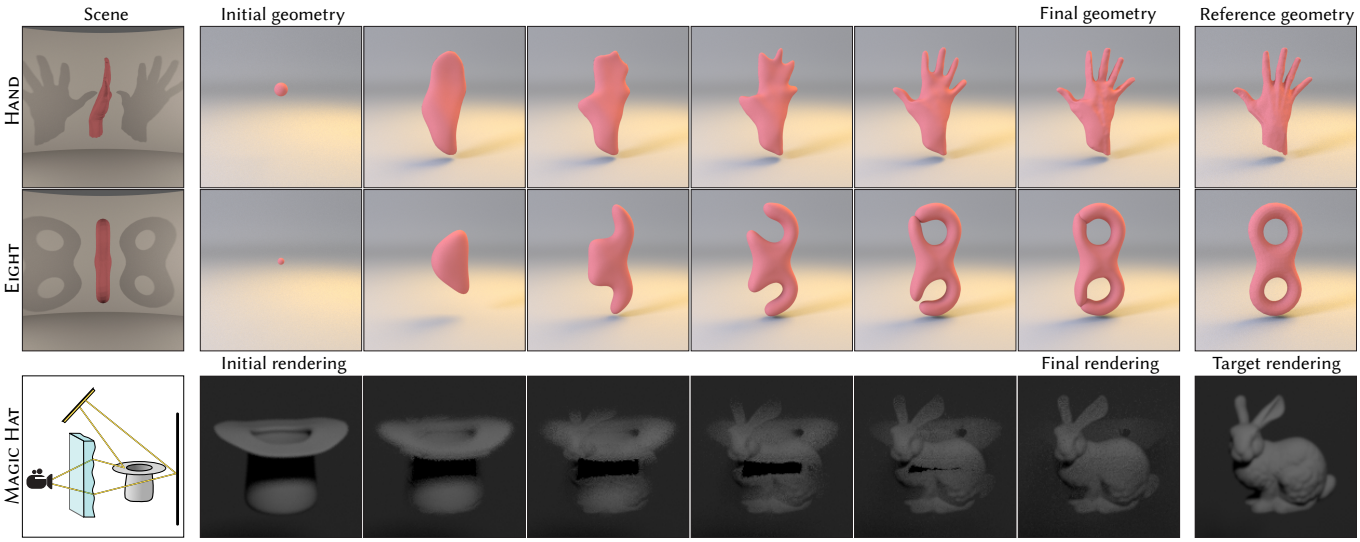


Fig. 15. **Optimization tasks involving discontinuous derivatives.** The first two experiments we reconstruct the shape of an object from a single-view capture. The last row optimizes the position of a hat and the height field of a refractive plate with a small amount of roughness (Beckmann $\alpha = 0.01$) to form an image of a bunny. (The supplemental video contains animated versions of the first two experiments.)

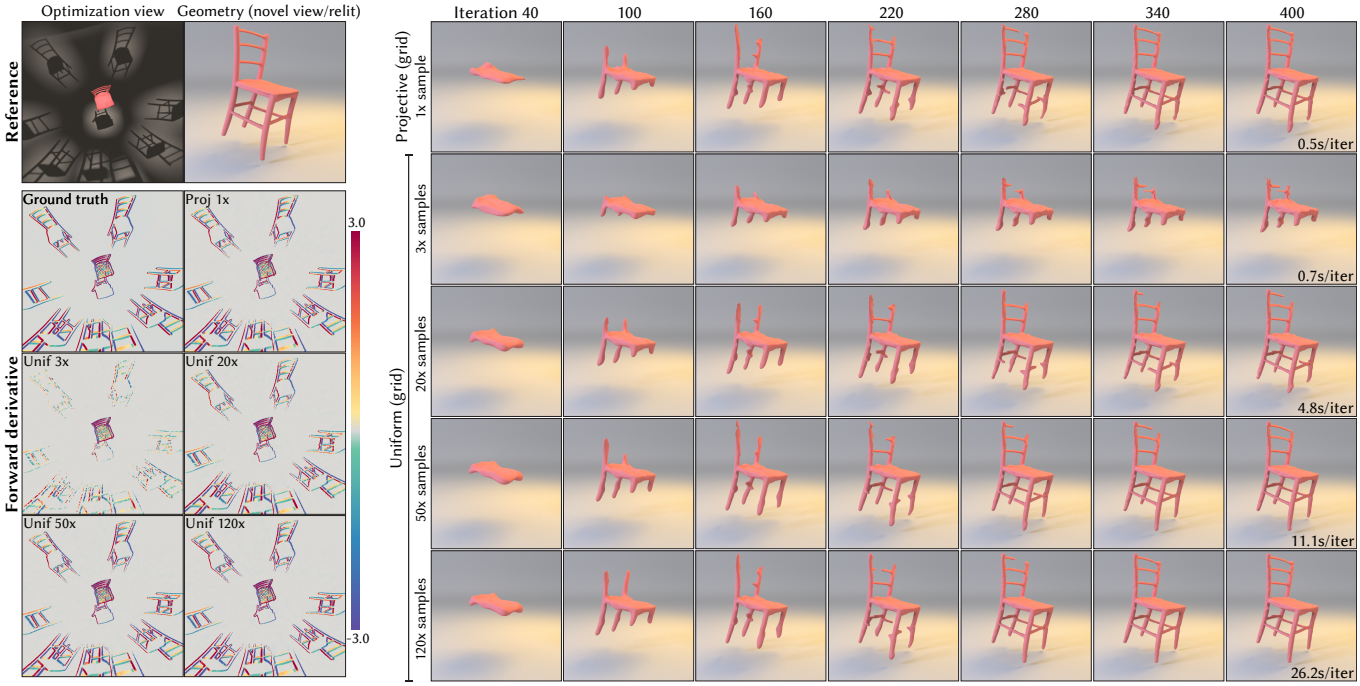


Fig. 16. **The effect of gradient quality on geometric reconstruction.** We reconstruct a triangle mesh from a single reference view with multiple shadows on a curved floor (upper left), comparing projective and uniform sampling for different iteration and sample counts. Timing values in the rightmost column quantify discontinuity-related costs while ignoring the shared overheads of primal rendering, preconditioned gradient descent [Nicolet et al. 2021], and re-meshing.

designed so that the shadows reveal a sufficient amount of information to disambiguate the shape and in principle enable convergence using gradient descent. We reuse the optimization strategy from Figure 15 and perform grid-based guiding with an equal resolution in all five optimization runs. The only difference lies in the initialization of the grid and the resulting gradient quality. We also visualize forward gradients for a horizontal displacement to explain the differences in convergence.

Optimization using low-quality gradients ("*uniform, 3x samples*") stagnates, while estimates with lower variance ("*uniform, 20x/50x/120x*

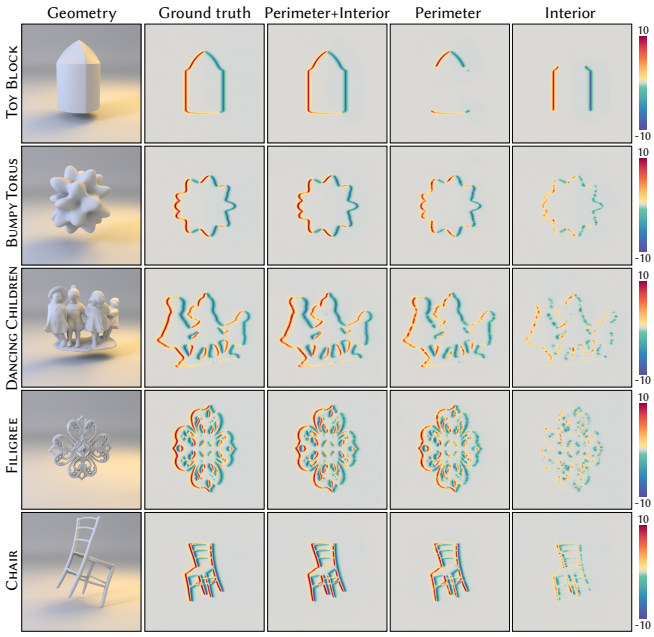


Fig. 17. This visualization shows visibility-induced derivatives of signed distance functions (SDFs), separating the contributions from perimeter and interior. These examples all use a 128^3 SDF grid with trilinear interpolation.

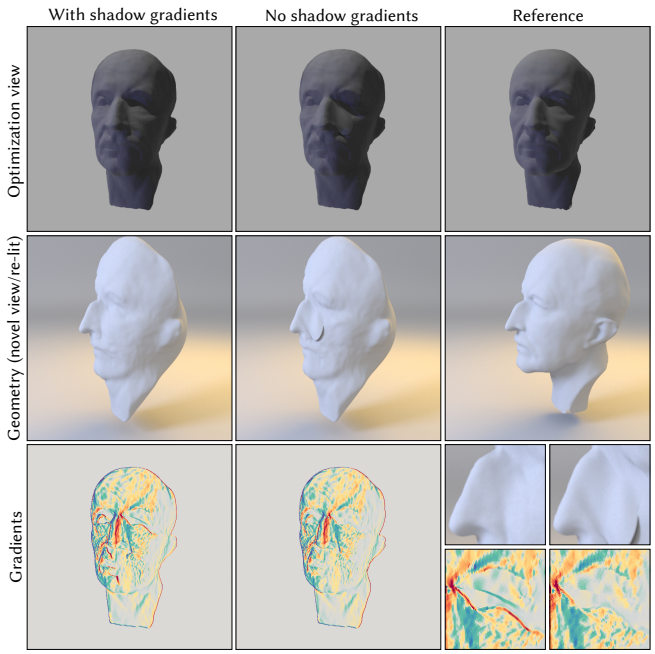


Fig. 18. Single-view reconstruction *with* and *without* derivatives due to self-shadowing. Accounting for them resolves ambiguity in this information-constrained scenario. The gradient image in the last row (with respect to a horizontal translation) illustrates the missing derivatives. (The supplemental video contains an animated version of this figure.)

samples") lead to progressively better convergence in an equal-iteration comparison (though this comes at a greatly increased computational cost). Our method ("*projective, 1x sample*") achieves a good balance between gradient quality and computation time.

Signed distance functions. Figure 17 validates the use of our local boundary integral formulation (Equation 4) for SDFs. We separate the derivative contributions of perimeter and interior, which superimpose to produce a gradient matching the ground truth.

Importance of indirect derivatives. Figure 18 shows the impact of indirectly observed discontinuities in a single-view mesh reconstruction. Such indirect effects are an inherent property of essentially any realistic image. When self-shadowing is not taken into account during the differentiation, the optimizer lacks the information that the shadow near the nose is caused by the eyebrow. It attempts to conceal the undesired shadow by distorting the nose to cover it. With self-shadowing derivatives computed by our method, the optimizer instead subtly adjusts the eyebrow. Other aspects (e.g. nose height) also improve, since shadows reduce ambiguities in the challenging single-view setting.

7 CONCLUSION

A central discovery of path-space differentiable rendering [Zhang et al. 2020] was that the interior and boundary derivatives decouple, allowing each part to be handled independently. In contrast, our work identifies substantial synergies between these two, indicating that a full separation is generally inadvisable. Our experiments demonstrate that the boundary term can greatly benefit from information gathered during the interior term's simulation.

This idea leads to a modular framework of projections and parameterizations that tie into the established architecture of contemporary rendering systems. Possible constructions include hopping from triangle to triangle, jumping over larger distances, and the numerical solution of nonlinear equations. We find this an interesting design space and believe that future work could also specifically target linear elements to overcome the difficult "blade of grass" case mentioned earlier.

The discontinuous nature of boundary sample space remains a significant source of inefficiency. Both prior work [Yan et al. 2022] and this article demonstrate that edge permutations and adaptive discretizations can mitigate this problem to a certain extent, but these are essentially just band-aids. In general, the representation is intrinsically smooth, but this property is lost in the mapping onto a parameter space. Truly smooth global parameterizations of this challenging domain are an interesting topic for future work.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 948846).

REFERENCES

Sai Praveen Bangaru, Michaël Gharbi, Fujun Luan, Tzu-Mao Li, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable rendering of neural sdfs through reparameterization. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* 39, 6 (2020), 1–18.

Algorithm 2. Pseudocode of the optimization loop and a vectorized reverse-mode implementation of the rendering function.

```
def optimize()
    for l in range(N_iter):
        loss = l2(render(scene), reference)
        loss.backward()
        optimizer.step(scene.grad)

@backward
def render(scene, grad_img):
    rays = scene.sensor.sample_rays()
    # Propagate derivative of the continuous part with PRB and
    # return path segments
    img_cont, segments = prb_backward(grad_img)
    shape = segments.shape

    # Project path segments, returns an array of new segments
    # and failure markers.
    proj_segments = shape.project(segments)

    # Convert into a 3D point in boundary sample space
    boundary_sample = shape.map_to_sample(proj_segments)

    # Evaluate the boundary integrand without motion
    value = eval_integrand(proj_segments)

    # Initialize the guiding data structure (the octree may
    # call eval_integrand internally)
    distr = guiding_octree(boundary_sample, value)

    # Draw samples from the guiding distribution
    boundary_sample, pdf = distr.sample_pdf()
    boundary_segment = shape.map_to_segment(boundary_sample)

    # Particle tracer style render starting from boundary segment
    img_boundary = render_boundary(boundary_segment, pdf)

    img_combined = img_boundary + img_cont
    img_combined.backward(grad_img)
```

Table 1. Guiding initialization time (in seconds) for Figure 10.

	EmptyCube	Bunny	Fertility	Filigree	Dragon
Uniform (5x)	0.52	0.49	0.52	0.46	0.47
Uniform (50x)	4.30	4.05	4.36	3.71	3.80
Projective	0.41	0.29	0.30	0.30	0.29

Table 2. Guiding initialization time (in seconds) for Figure 11.

	BumpyTorus	Botijo	Neptune
Adaptive quadrature	1.12	1.11	1.11
Projective (grid)	0.39	0.26	0.26
Projective (octree)	0.73	0.48	0.46

Table 3. Guiding initialization time (in seconds) for Figure 14.

	Knot	Coil	Winding	Sausage	KnittingYarn
Uniform (grid, 5x)	0.75	0.68	1.92	0.98	0.59
Uniform (grid, 50x)	7.22	6.52	18.98	9.68	5.48
Projective (grid)	0.13	0.11	0.30	0.12	0.14
Projective (octree)	0.26	0.26	0.64	0.30	0.33

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically differentiating parametric discontinuities. *ACM Trans. Graph.* 40, 4 (2021), 1–18.

Thomas Booth. 2007. Unbiased Monte Carlo Estimation of the Reciprocal of an Integral. *Nuclear Science and Engineering* 156 (07 2007), 403–407.

David Bremer and John F. Hughes. 1998. Rapid Approximate Silhouette Rendering of Implicit Surfaces. In *Proceesings of Implicit Surfaces 98.*

Forrester Cole, Kyle Genova, Avneesh Sud, Daniel Vlasic, and Zhoutong Zhang. 2021. Differentiable surface rendering via non-differentiable sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 6088–6097.

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

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

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

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

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

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6 (2018), 1–11.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. *ACM Trans. Graph.* 39, 6 (nov 2020).

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

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *European Conference on Computer Vision (ECCV).* Springer, 154–169.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* 38, 6 (2019), 1–14.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021).

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

Marios Papas, Thomas Houit, Derek Nowrouzezahrai, Markus Gross, and Wojciech Jarosz. 2012. The Magic Lens: Refractive Steganography. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (Nov. 2012).

Felix Petersen, Bastian Goldluecke, Christian Borgelt, and Oliver Deussen. 2022. GenDR: A Generalized Differentiable Renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 4002–4011.

Osborne Reynolds. 1903. *The sub-mechanics of the universe.* Vol. 3. University Press.

Herman Hansson Söderlund, Alex Evans, and Tomas Akenine-Möller. 2022. Ray Tracing of Signed Distance Function Grids. *Journal of Computer Graphics Techniques Vol* 11, 3 (2022).

T. S. Trowbridge and K. P. Reitz. 1975. Average irregularity representation of a rough surface for ray reflection. *J. Opt. Soc. Am.* 65, 5 (May 1975), 531–536.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph.D. Dissertation. Stanford University.

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques.* Association for Computing Machinery.

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

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022).

Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph.* 41, 4 (2022), 1–13.

Yuting Yang, Connelly Barnes, Andrew Adams, and Adam Finkelstein. 2022. A δ: autodiff for discontinuous programs-applied to shaders. *ACM Trans. Graph.* 41, 4 (2022), 1–24.

Cheng Zhang, Bailey Miller, Kan Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-space differentiable rendering. *ACM Transactions on Graphics* 39, 4 (2020).

Yang Zhou, Lifan Wu, Ravi Ramamoorthi, and Ling-Qi Yan. 2021. Vectorization for Fast, Analytic, and Differentiable Visibility. *ACM Transactions on Graphics* 40, 3 (July 2021).