Path-Space Differentiable Rendering of Implicit Surfaces

Siwei Zhou
siwei.zhou@ist.osaka-u.ac.jp
Osaka University
Japan

Youngha Chang
ychang@tcu.ac.jp
Tokyo City University
Japan

Nobuhiko Mukai
nmukai@tcu.ac.jp
Tokyo City University
Japan

Hiroaki Santo
santo.hiroaki@ist.osaka-u.ac.jp
Osaka University
Japan

Fumio Okura
okura@ist.osaka-u.ac.jp
Osaka University
Japan

Yasuyuki Matsushita
yasumat@ist.osaka-u.ac.jp
Osaka University
Japan

Shuang Zhao
shz@ics.uci.edu
UC Irvine & NVIDIA
USA

(a) Ordinary

(b) Derivative

Figure 1: We apply path-space differentiable rendering to implicit surfaces. Specifically, we derive the necessary mathematical tools and then propose Monte Carlo estimators of the boundary integrals. In this example, we showcase our method can handle the boundary integrals even with complex shapes.

ABSTRACT

Physics-based differentiable rendering is a key ingredient for integrating forward rendering into probabilistic inference and machine learning pipelines. As a state-of-the-art formulation for differentiable rendering, differential path integrals have enabled the development of efficient Monte Carlo estimators for both interior and boundary integrals. Unfortunately, this formulation has been designed mostly for explicit geometries like polygonal meshes.

In this paper, we generalize the theory of differential path integrals to support implicit geometries like level sets and signed-distance functions (SDFs). In addition, we introduce new Monte Carlo estimators for efficiently sampling discontinuity boundaries that are also implicitly specified. We demonstrate the effectiveness of our theory and algorithms using several differentiable-rendering and inverse-rendering examples.

CCS CONCEPTS

• **Computing methodologies → Rendering**.

KEYWORDS

Differentiable rendering, stochastic sampling

ACM Reference Format:
Siwei Zhou, Youngha Chang, Nobuhiko Mukai, Hiroaki Santo, Fumio Okura, Yasuyuki Matsushita, and Shuang Zhao. 2024. Path-Space Differentiable Rendering of Implicit Surfaces. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657473

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657473

1 INTRODUCTION

Physics-based rendering is the core technology for high quality computer-generated imagery such as those seen in feature films. Physics-based differentiable rendering brings this technology to other fields such as machine learning. The main object being studied in this field is the boundary integrals. Those boundary integrals occur when shape is to be differentiated with respect to. Early research focused on deriving these boundary integrals in different contexts [Li et al. 2018; Zhang et al. 2019; Loubet et al. 2019; Zhang et al. 2020, 2021]. Recent works focus more on how to efficiently estimate those boundary integrals [Yan et al. 2022; Xu et al. 2023; Zhang et al. 2023].

A common assumption in those researches is the representation of the shape being mesh or parametric surfaces. While mesh is a popular shape representation, another popular one is implicit surface. An implicit surface is defined by the zero-level set of some scalar-valued function, i.e., where the function takes its value zero.

In practice, there are two families of Monte Carlo estimators to handle the boundary integrals, namely, reparameterization [Loubet et al. 2019; Bangaru et al. 2020; Xu et al. 2023] and edge-sampling [Li et al. 2018; Zhang et al. 2020, 2023]. Reparameterization replaces the boundary integrals with equivalent interior ones and, thus, avoids explicitly sampling discontinuities boundaries. In addition, these techniques can be adopted for implicit geometries relatively easily [Vicini et al. 2022; Bangaru et al. 2022]. On the other hand, the reparameterized interior integrals remain difficult to importance sample, which can cause the reparameterization-based methods to suffer from higher variance.

On the other hand, edge-sampling-based methods directly estimate the boundary integrals using Monte Carlo integration. When importance sampled properly [Zhang et al. 2023], they tend to produce lower-variance estimates than the reparameterization-based methods. Unfortunately, state-of-the-art techniques in this category—such as the path-space formulation by Zhang et al. [2020]—have been mostly derived for explicit geometries. For example, Monte Carlo sampling over silhouettes of implicit surfaces has not been addressed by prior methods.

In this paper, we bridge this gap by introducing a path-space theory for differentiable rendering of implicit surfaces. Concretely, Our contributions include:

- Generalizing the path-space differentiable rendering framework [Zhang et al. 2020] to include implicit surfaces (§4);
- Introducing Monte Carlo estimators for the boundary integrals with implicit surfaces (§5).

We demonstrate the effectiveness of our method using several synthetic differentiable-rendering and inverse-rendering examples.

2 RELATED WORKS

Forward rendering of implicit surfaces. Implicit surface rendering or isosurface rendering can be used to visualized volumetric data such as those obtained by CT or MRI. One way to render implicit surfaces is to triangulated the isosurface using Marching Cubes [Lorensen and Cline 1998] and then render the triangular mesh using regular rendering methods. One can also render the implicit surface from point samples generated by particle repulsion [Witkin and Heckbert 1994]. Yet another way is to use ray tracing [Barr 1986; Levoy 1988]. Ray tracing requires a ray-surface intersection routine, examples of which include sphere tracing [Hart 1996] and interval analysis [Flórez et al. 2008]. Besides the above surface rendering methods, volume rendering [Drebin et al. 1988] can also be used to render implcit surfaces.

Table 1: Commonly used symbols in this paper. The rightmost column indicates π-dependency.

Symbol	Definition	π-dep.
$\mathcal{M}$	The union of all surfaces in the scene	Yes
$\mathcal{B}$	Reference surface	**No**
$\mathsf{X}(\cdot, \pi)$	Differentiable one-to-one mapping transforming $\mathcal{B}$ to $\mathcal{M}(\pi)$	Yes
$\hat{\Omega}$	material path space	**No**
$\partial\hat{\Omega}$	material boundary path space	Yes
$\hat{f}$	material measurement function	Yes
$\Delta\mathcal{B}_K$	Jump discontinuities of $\hat{f}$ with respect to $\boldsymbol{p}_K$	Yes
$\boldsymbol{n}_\partial$	Normal of discontinuity curve	Yes
$\boldsymbol{v}$	Velocity field over $\mathcal{B}$ defined in Eq. (15)	Yes
V_∂	Curve normal velocity defined in Eq. (8)	Yes

Physics-based differentiable rendering. Physics-based differentiable rendering correctly accounts for derivatives of rendered images with respect to shapes. Early works [Li et al. 2018; Loubet et al. 2019; Zhang et al. 2019; Bangaru et al. 2020; Zhang et al. 2020, 2021] derive the exact mathematical form of the derivatives in various contexts. Following works [Xu et al. 2023; Yan et al. 2022; Yu et al. 2022, 2023; Zhang et al. 2023; Vicini et al. 2021; Nimier-David et al. 2020] attempt to estimate the derivatives more efficiently. An common assumption shared by these works is the underlying surface representation being mesh or parametric surface.

Differentiable rendering of implicit surfaces. Bangaru et al. [2020]'s method despite being designed for meshes, can be easily applied to implicit surfaces. Bangaru et al. [2022] and Vicini et al. [2022] propose more efficient methods to specifically do differentiable rendering on signed distance functions. Our method applies to general implicit surfaces defined as the zero-level set of some function.

3 PRELIMINARIES

We now briefly revisit some mathematical and algorithmic preliminaries related to the differential path integral formulation [Zhang et al. 2020]. Table 1 presents a list of symbols that are commonly used in this paper and their definitions.

Path integral. In forward rendering, the path integral formulation introduced by Veach [1997] has allowed the development of many advanced Monte Carlo techniques (*e.g.*, bidirectional path tracing). Under this formulation, the response of a radiometric detector is expressed as a *path integral*:

$$I = \int_\Omega f(\bar{x}) \, \mathrm{d}\mu(\bar{x}), \tag{1}$$

where $\Omega := \cup_{N\geq 1}\mathcal{M}^{N+1}$ is the *path space* comprising *light transport paths* $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ with $\mathcal{M}$ being the union of all object surfaces, f is the *measurement contribution* function, and μ is the corresponding area-product measure.

Material-form reparameterization. Differentiating Eq. (1) with respect to some arbitrary parameter $\pi \in \mathbb{R}$ can be complicated when the scene geometry $\mathcal{M}$—and hence the path space Ω—evolves with this parameter. To address this problem, Zhang et al. [2020] have proposed to reparameterize the path integral (1) using some fixed *reference surface* $\mathcal{B}$ coupled with a differentiable one-to-one mapping $\mathsf{X}(\cdot, \pi)$ that transforms the fixed reference surface $\mathcal{B}$ to the evolving surface $\mathcal{M}(\pi)$ for all π.

This mapping allows a change of variable from light paths $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ to *material paths* $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ with $\boldsymbol{x}_n = \mathsf{X}(\boldsymbol{p}_n, \pi)$ for all $0 \leq n \leq N$. Applying this change of variable to Eq. (1) yields the *material-form path integral*:

$$I = \int_{\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{2}$$

which is over the *material path space* $\hat{\Omega} = \cup_{N\geq 1}\mathcal{B}^{N+1}$. Further, the integrand of this path integral $\hat{f}$ is the *material measurement contribution* defined as

$$\hat{f}(\bar{\boldsymbol{p}}) = f(\bar{\boldsymbol{x}}) \prod_{n=0}^{N} J(\boldsymbol{p}_n, \pi), \tag{3}$$

where

$$J(\boldsymbol{p}, \pi) := \|\mathrm{d}A(\boldsymbol{x})/\mathrm{d}A(\boldsymbol{p})\| \tag{4}$$

is the Jacobian resulting from the reparameterization with A indicating the surface-area measure.

Differential path integral. Zhang et al. [2020] have shown that the derivative of Eq. (2) with respect to an arbitrary parameter π can be expressed as material-form *differential path integrals* of the form

$$\frac{\mathrm{d}I}{\mathrm{d}\pi} = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}}{\mathrm{d}\pi}\hat{f}(\bar{\boldsymbol{p}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\, V_\partial(\boldsymbol{p}_K)\, \mathrm{d}\dot{\mu}}_{\text{boundary}}, \tag{5}$$

where the integrand of the *interior* term is the derivative of the material measurement contribution $\hat{f}$ defined in Eq. (3).

The *boundary* integral in Eq. (5) is over the *material boundary path space* $\partial\hat{\Omega}$ comprising *material boundary paths*

$$\bar{\boldsymbol{p}}_K = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N), \quad \boldsymbol{p}_K \in \Delta\mathcal{B}(\boldsymbol{p}_{K-1}), \tag{6}$$

containing exactly one vertex $\boldsymbol{p}_K$ constrained to a set of visibility boundaries $\Delta\mathcal{B}(\boldsymbol{p}_{K-1})$. Precisely, $\boldsymbol{p}_K$ is a jump discontinuity point of the *mutual visibility* $\mathbb{V}(\mathsf{X}(\boldsymbol{p}_{K-1}, \pi) \leftrightarrow \mathsf{X}(\cdot, \pi))$ with $\boldsymbol{p}_{K-1}$ fixed. We define boundary segments to be the pair

$$(\boldsymbol{p}_{K-1}, \boldsymbol{p}_K) \in \mathcal{B} \times \Delta\mathcal{B}(\boldsymbol{p}_{K-1}), \tag{7}$$

and refer to them as $\overline{\boldsymbol{p}_{K-1}\boldsymbol{p}_K}$.

Additionally, V_∂ captures the rate at which the visibility boundary shifts (with respect to π) and is defined as

$$V_\partial(\boldsymbol{p}_K) = \boldsymbol{n}_\partial(\boldsymbol{p}_K) \cdot \frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\pi}, \tag{8}$$

where $\boldsymbol{n}_\partial(\boldsymbol{p}_K)$ denotes the unit normal of the visibility boundary (that is perpendicular to both the surface normal $\boldsymbol{n}_\mathcal{B}(\boldsymbol{p}_K)$ and the tangent of the curve). We assume without loss of generality that $\boldsymbol{n}_\partial(\boldsymbol{p}_K)$ points toward the occluded side of the visibility boundary.

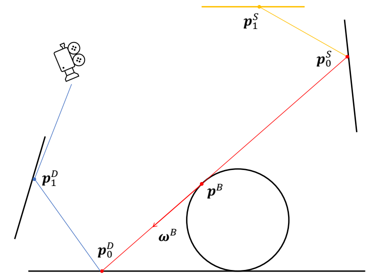


Figure 2: Boundary segments: A boundary light path (6) is comprised of a boundary segment (7) $\overline{\boldsymbol{p}_0^\mathrm{S}\,\boldsymbol{p}_0^\mathrm{D}}$ (shown in red) as well as a source and a detector subpath (illustrated in orange and blue, respectively). The two endpoints of the boundary segment (7) are located on the visibility boundary of each other. A boundary segment (6) $\overline{\boldsymbol{p}_0^\mathrm{S}\,\boldsymbol{p}_0^\mathrm{D}}$ can be uniquely determined with a surface point $\boldsymbol{p}^\mathrm{B} \in \mathcal{B}$ interior to the segment, along with a direction ω^B.

Multi-directional form of the boundary path integral. Numerically estimating the *boundary* integral in Eq. (5) is known to be challenging due to the need of sampling boundary segments which requires silhouette detection.

To avoid silhouette detection at every vertex, Zhang et al. [2020] propose to re-index a material boundary path as

$$\bar{\boldsymbol{p}} = (\boldsymbol{p}_s^\mathrm{S}, \ldots, \boldsymbol{p}_0^\mathrm{S}, \boldsymbol{p}_0^\mathrm{D}, \ldots, \boldsymbol{p}_t^\mathrm{D}) \tag{9}$$

with $\overline{\boldsymbol{p}_0^\mathrm{S}\boldsymbol{p}_0^\mathrm{D}}$ being the boundary segment that further separates the full path $\bar{\boldsymbol{p}}$ into a *source subpath* $(\boldsymbol{p}_s^\mathrm{S}, \ldots, \boldsymbol{p}_0^\mathrm{S})$ and a *detector subpath* $(\boldsymbol{p}_0^\mathrm{D}, \ldots, \boldsymbol{p}_t^\mathrm{D})$. Then, the *boundary* component of Eq. (5) can be rewritten in a *multi-directional form* as

$$\int_{\mathcal{B}} \int_{\Delta\mathcal{B}(\boldsymbol{p}_{K-1})} \left(\int_{\hat{\Omega}} \hat{f}^\mathrm{S}\, \mathrm{d}\mu(\bar{\boldsymbol{p}}^\mathrm{S})\right) \hat{f}^\mathrm{B} \left(\int_{\hat{\Omega}} \hat{f}^\mathrm{D}\, \mathrm{d}\mu(\bar{\boldsymbol{p}}^\mathrm{D})\right) \mathrm{d}\ell(\boldsymbol{p}_0^\mathrm{D})\, \mathrm{d}A(\boldsymbol{p}_0^\mathrm{S}), \tag{10}$$

where $\hat{f}^\mathrm{B}$, $\hat{f}^\mathrm{S}$, and $\hat{f}^\mathrm{D}$ are components of the integrand $\hat{f}(\bar{\boldsymbol{p}})\, V_\partial(\boldsymbol{p}_0^\mathrm{D})$ capturing the contributions of the boundary segment, the source subpath, and the detector subpath, respectively.

Using the multi-directional-form integral (10), one can construct a boundary path by (i) sampling the material boundary segment $\overline{\boldsymbol{p}_0^\mathrm{S}\,\boldsymbol{p}_0^\mathrm{D}}$; and (ii) using standard methods such as unidirectional and bidirectional path sampling to build the source and detector subpaths. This avoids the need of expensive silhouette detection [Li et al. 2018] and, thus, scales significantly better to scenes with detailed geometries.

Previously, Zhang et al. [2020] have derived the exact form of Eq. (10) and the corresponding sampling procedures for explicit geometries (i.e., polygonal meshes). In this paper, we will derive the multi-directional-form *boundary* path integral for implicit geometries and develop new Monte Carlo sampling methods to efficiently estimate this integral.

4 DIFFERENTIAL PATH INTEGRALS FOR IMPLICIT SURFACES

In what follows, we explain how the formulation of differential path integrals can be realized for implicit surfaces.

Specifically, we first discuss how the material-form parameterization can be defined for implicit surfaces in §4.1. Then, we derive the multi-directional form of the *boundary* path integral—which is crucial for efficient estimation of this term—in §4.2.

To simplify the notations, we assume without loss of generality that all derivatives are to be evaluated at $\pi = 0$ and define

$$\partial_\pi h \coloneqq \left[\frac{\mathrm{d}h}{\mathrm{d}\pi}\right]_{\pi=0} \tag{11}$$

for all h.

4.1 Material-Form Parameterization

In this paper, we focus on differentiable rendering of evolving closed implicit surfaces defined as zero-level sets of some scalar-valued function $\phi(\boldsymbol{x};\pi)$:

$$\mathcal{M}(\pi) \coloneqq \{\boldsymbol{x} \in \mathbb{R}^3 \,:\, \phi(\boldsymbol{x};\pi) = 0\}. \tag{12}$$

To facilitate the differentiation of the path integral, we follow Zhang et al. [2020] by reparameterizing the evolving implicit surface $\mathcal{M}(\pi)$ using a closed reference surface $\mathcal{B}$ that is independent of the parameter π and defined as

$$\mathcal{B} \coloneqq \mathcal{M}(0) = \{\boldsymbol{x} \in \mathbb{R}^3 \,:\, \phi(\boldsymbol{x};0) = 0\}. \tag{13}$$

Unfortunately, unlike when using explicit geometries, it is hard—if not impossible—to define a one-to-one mapping $\mathsf{X}(\cdot,\pi)$ that transforms the reference surface $\mathcal{B}$ to the evolving $\mathcal{M}(\pi)$ for all π. Instead, we define a mapping illustrated in Figure 3 that is only one-to-one *at* $\pi = 0$ via

$$\mathsf{X}(\boldsymbol{p},\pi) \coloneqq \boldsymbol{p} + t(\boldsymbol{p},\pi)\,\boldsymbol{n}_{\mathcal{B}}(\boldsymbol{p}), \tag{14}$$

where:

- $\boldsymbol{n}_{\mathcal{B}}(\boldsymbol{p})$ denotes the unit normal of the reference surface $\mathcal{B}$ at $\boldsymbol{p}$ and is assumed to be smooth and differentiable;
- $t(\boldsymbol{p},\pi) \in \mathbb{R}$ indicates the distance from $\boldsymbol{p}$ to the first intersection between the (fixed) line passing through $\boldsymbol{p}$ via the direction $\boldsymbol{n}_{\mathcal{B}}(\boldsymbol{p})$ and the evolving surface $\mathcal{M}(\pi)$.

$\mathsf{X}(\cdot,0)$ is the identity map and thus one-to-one.

We define the velocity of $\boldsymbol{p} \in \mathcal{B}$ as the derivative of X w.r.t. π:

$$\boldsymbol{v}(\boldsymbol{p}) \coloneqq (\partial_\pi t)(\boldsymbol{p})\,\boldsymbol{n}_{\mathcal{B}}(\boldsymbol{p}), \tag{15}$$

and with our definition of $\mathcal{B}$ and $\pi = 0$ [Stam and Schmidt 2011] [Yariv et al. 2020],

$$(\partial_\pi t)(\boldsymbol{p}) = -\frac{\partial_\pi \phi(\boldsymbol{p})}{\|\nabla_{\boldsymbol{p}}\phi(\boldsymbol{p})\|}. \tag{16}$$

Recall that $J(\boldsymbol{p},\pi) \coloneqq \|\mathrm{d}A(\boldsymbol{x})/\mathrm{d}A(\boldsymbol{p})\|$ is the ratio between surface elements at $\boldsymbol{x} = \mathsf{X}(\boldsymbol{p},\pi) \in \mathcal{M}(\pi)$ and $\boldsymbol{p} \in \mathcal{B}$. It holds that

$$(\partial_\pi J)(\boldsymbol{p}) \coloneqq \left[\frac{\mathrm{d}}{\mathrm{d}\pi} J(\boldsymbol{p},\pi)\right]_{\pi=0} = \kappa(\boldsymbol{p})\,V(\boldsymbol{p}). \tag{17}$$

We leave the proof in the supplemental.

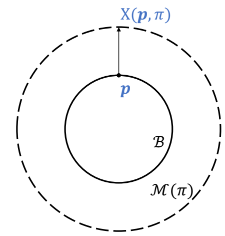


Figure 3: $\mathsf{X}(\boldsymbol{p},\pi)$ is the intersection of $\boldsymbol{p}$'s normal with the evolving surface $\mathcal{M}(\pi)$.

4.2 Deriving Boundary Segment

Sampling the boundary segment as in Eq. (10) still requires silhouette detection. Zhang et al. [2020] performs a change of integration variable to avoid the silhouette detection *with meshes* for most types of boundary segment except for those with camera as one of its endpoints. We derive the change of integration variable for higher-order smooth surfaces. Note Zhang et al. [2023] also derive the same change of integration variable in a concurrent work. We provide a high-level overview here and leave the complete derivation in the supplemental.

For brevity, we name the integrand of Eq. (10) F and drop the subscripts and start from there:

$$\int_{\mathcal{B}} \int_{\Delta\mathcal{B}} F(\boldsymbol{p}^{\mathsf{S}}, \boldsymbol{p}^{\mathsf{D}})\,\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{D}})\,\mathrm{d}A(\boldsymbol{p}^{\mathsf{S}}). \tag{18}$$

Our goal is to change the integration variable from $(\boldsymbol{p}^{\mathsf{S}}, \boldsymbol{p}^{\mathsf{D}})$ to $(\boldsymbol{p}^{\mathsf{B}}, \boldsymbol{\omega}^{\mathsf{B}})$ illustrated in Figure 4.

$\boldsymbol{p}^{\mathsf{D}}$ are silhouette projected onto surface, and we first project them back onto the silhouette $\boldsymbol{p}^{\mathsf{B}}$. The projection results in the following change of variable:

$$\int_{\mathcal{B}} \int_{\Sigma\mathcal{B}} F(\boldsymbol{p}^{\mathsf{S}}, \boldsymbol{p}^{\mathsf{D}})\,\frac{\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{D}})}{\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{B}})}\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{B}})\,\mathrm{d}A(\boldsymbol{p}^{\mathsf{S}}), \tag{19}$$

where $\Sigma\mathcal{B}$ denotes the silhouette of $\mathcal{B}$ as seen from the perspective of $\boldsymbol{p}^{\mathsf{S}}$.

$\boldsymbol{p}^{\mathsf{S}}$ can vary in two directions, namely $\mathrm{d}\ell_n$ and $\mathrm{d}\ell_t$, where $\mathrm{d}\ell_n$ is the projection of $\boldsymbol{\omega}^{\mathsf{B}}$ onto the tangent plane at $\boldsymbol{p}^{\mathsf{S}}$ and $\mathrm{d}\ell_t$ is the direction orthogonal to $\mathrm{d}\ell_n$. When $\boldsymbol{p}^{\mathsf{S}}$ varies along $\mathrm{d}\ell_n$, $\boldsymbol{p}^{\mathsf{B}}$ varies along $\boldsymbol{\omega}^{\mathsf{B}}$, so we can map the variation of $\boldsymbol{p}^{\mathsf{S}}$ along $\mathrm{d}\ell_n$ onto that of $\boldsymbol{p}^{\mathsf{B}}$ along $\boldsymbol{\omega}^{\mathsf{B}}$. The mapping results in the following change of variable:

$$\int_{\mathcal{B}} \int_{\Gamma\mathcal{B}} F(\boldsymbol{p}^{\mathsf{S}}, \boldsymbol{p}^{\mathsf{D}})\,\frac{\mathrm{d}\ell_n(\boldsymbol{p}^{\mathsf{S}})\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{D}})}{\mathrm{d}\ell_n(\boldsymbol{p}^{\mathsf{B}})\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{B}})}\mathrm{d}\ell_t(\boldsymbol{p}^{\mathsf{S}})\,\mathrm{d}A(\boldsymbol{p}^{\mathsf{B}}), \tag{20}$$

where $\Gamma\mathcal{B}$ denotes the intersection of $\mathcal{B}$ and the tangent plane at $\boldsymbol{p}^{\mathsf{B}}$.

$\boldsymbol{p}^{\mathsf{S}}$ is the intersection of $\boldsymbol{p}^{\mathsf{B}}$'s tangent vector $\boldsymbol{\omega}^{\mathsf{B}}$ with $\mathcal{B}$, meaning there is a one-to-one mapping between $\boldsymbol{p}^{\mathsf{S}}$ and $\boldsymbol{\omega}^{\mathsf{B}}$ defined by the intersection. we use this mapping to map $\boldsymbol{p}^{\mathsf{S}}$'s other variation onto $\boldsymbol{\omega}^{\mathsf{B}}$'s. And the mapping results in the following change of variable:

$$\int_{\mathcal{B}} \int_{\mathbb{S}} F(\boldsymbol{p}^{\mathsf{S}}, \boldsymbol{p}^{\mathsf{D}})\,\frac{\mathrm{d}\ell_t(\boldsymbol{p}^{\mathsf{S}})\mathrm{d}\ell_n(\boldsymbol{p}^{\mathsf{S}})\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{D}})}{\mathrm{d}\theta(\boldsymbol{\omega}^{\mathsf{B}})\mathrm{d}\ell_n(\boldsymbol{p}^{\mathsf{B}})\mathrm{d}\ell(\boldsymbol{p}^{\mathsf{B}})}\mathrm{d}\theta(\boldsymbol{\omega}^{\mathsf{B}})\,\mathrm{d}A(\boldsymbol{p}^{\mathsf{B}}). \tag{21}$$

We leave the detailed derivation in the supplemental.

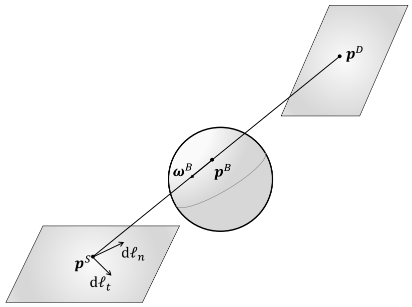


Figure 4: We change the integration variable from $(\boldsymbol{p}^{\mathrm{S}}, \boldsymbol{p}^{\mathrm{D}})$ to $(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$.

5 OUR MONTE CARLO ESTIMATORS

We use Monte Carlo integration to estimate the differential integrals of Eq. (5). In practice, the interior integral in Eq. (5) can be easily computed through the differentiation of the forward rendering process. However, acquiring the boundary integrals is not straightforward since the domain of those boundary integrals is hard to sample efficiently. To address this problem with implicit surfaces, we introduce new Monte Carlo estimators for the boundary integrals.

There are two kinds of boundary integrals. Aforementioned Eq. (21) is called the secondary boundary integral whose boundary segment has surface points as its two endpoints. On the other hand, Eq. (10) with $\boldsymbol{p}_0^{\mathrm{S}}$ being a camera is called the primary boundary integral whose boundary segment has the camera as one of its endpoints. The change of integration variables in Section 4.2 helps to avoid silhouette detection, i.e., finding and sampling points from the silhouette of the surface from the point of view of some point, when sampling the boundary segment. However, when a camera is one of the endpoints, the change of variable no longer holds. And we have to perform silhouette detection for the camera to estimate the primary boundary integral.

We estimate the primary boundary integral by sampling points directly from the silhouette called the *primary edges* of the implicit surface. Estimating the primary boundary integral and sampling primary edges is easy with explicit geometries such as mesh. With mesh, one only needs to project the edges onto the image plane and sample from them. However, with implicit surfaces, we don't have a finite set of edges at our disposal or a parameterization of the primary edges. We instead use range analysis [Stolfi and De Figueiredo 1997] to obtain a tight bound and then utilize a proxy edge.

We estimate the secondary boundary integral by first sampling a surface point and then a tangent direction. We do not have a parameterization of the implicit surface, so we instead use a proxy surface bounding the surface. As for the tangent direction, we use next-event estimation to importance sample it.

5.1 Sampling Primary Edges

Discontinuities in the camera importance function form the primary edges of the surface. Primary edges is where the surface normal is perpendicular to the view vector, i.e., where the dot product of these two vectors equals zero, illustrated in Figure 5-(a). That dot product naturally defines another implicit surface we call the orientation function associated with the surface and camera, defined as follows:

$$\psi_{\phi,\boldsymbol{c}}(\boldsymbol{p}) := \langle \nabla\phi(\boldsymbol{p}), \mathrm{v}_{\boldsymbol{c}}(\boldsymbol{p})\rangle, \tag{22}$$

where $\boldsymbol{c}$ is the camera position and $\mathrm{v} := \boldsymbol{c} - \boldsymbol{p}$ is the view vector. We define the primary edges $\mathcal{S}$ to be the intersection of ϕ and ψ, visualized in Figure 5-(a), as follows:

$$\mathcal{S} := \{\boldsymbol{p} \mid \phi(\boldsymbol{p}) = 0, \psi_{\phi,\boldsymbol{c}}(\boldsymbol{p}) = 0\}. \tag{23}$$

We sample points from $\mathcal{S}$ by marching rays over some proxy surface as in Figure 5-(b). Let us use the surface's bounding box as an example. The projection of $\mathcal{S}$ on one of the faces of the bounding box is some curve on the face. Let us pretend this projection is one-to-one for now. Then sampling $\mathcal{S}$ is equivalent to sampling the projection. However, sampling the projection is still not tractable. So let us simplify the problem once more. Let us project the projection again onto some proxy curve we know how to sample. In this example, that is one of the edges of the face. Then we can sample a point on $\mathcal{S}$ by first sampling a point on the edge and then mapping this point onto $\mathcal{S}$, illustrated in Figure 5-(b).

Specifically, after sampling a point on the edge, we march along the direction perpendicular to the edge starting from this point. At each step, we shoot a ray in the direction normal to the face toward the surface. We evaluate ψ at the intersection and compare the value with that from the last step. If the signs are different, we take this intersection as the sample point on $\mathcal{S}$. This procedure is written in pseudocode in Algorithm 1.

In general, the ray might intersect either surface multiple times. That means we have to keep track of all the intersections from the previous step and keep marching all the way to the other edge. We instead use range analysis [Stolfi and De Figueiredo 1997] to bound $\mathcal{S}$ tighter and assume there is only one intersection in this tighter bound.

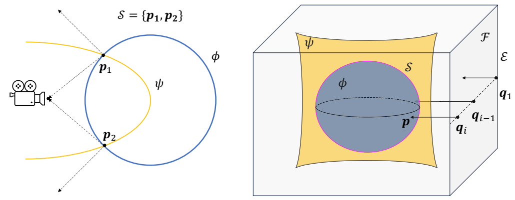
Range analysis [Stolfi and De Figueiredo 1997] is a method to propagate bounds through functions. We specifically use affine arithmetic. Given an axis-aligned bounding box (AABB), affine arithmetic tells us a bound of the function in this AABB. If the output bound does not contain zero, then this AABB is guaranteed not to intersect with the surface.

We cut the local space of the implicit surface into AABBs of the same size. The exact size is a hyperparameter and we use 6.1e^{-5}, i.e., cutting 42 times a unit cube in half, in all our experiments. After the cutting, we apply affine arithmetic to each AABB against ϕ and ψ. We throw away those AABBs guaranteed not to intersect with both surfaces and keep the others that possibly intersect both of the surfaces. Those AABBs bound $\mathcal{S}$ as in Figure 6.

5.2 Sampling Secondary Edges

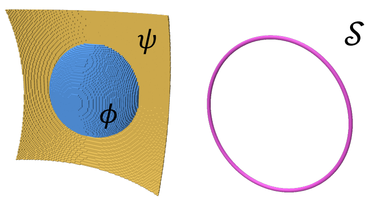
We sample the secondary boundary integral by first sampling a surface point and then a tangent direction.

5.2.1 Sampling Implicit Surfaces. The difficulty of sampling implicit surfaces is our surface representation is implicit, meaning we do not have a global parameterization. Our solution is to use some simple proxy surface amenable to parameterization.



(a) (b)

Figure 5: (a) Primary edges is the intersection of ϕ and ψ. In 2D, ϕ and ψ are curves, and $\mathcal{S}$ is a finite set of points. (b) We sample the primary edges by marching over a proxy surface



(a) (b)

Figure 6: AABBs classified by affine arithmetic as possibly intersect with (a) ϕ or ψ and (b) both of them. We bound silhouette $\mathcal{S}$ by (b) AABBs that possibly intersect with both the implicit surface ϕ and its associated orientation function ψ.

Imagine our implicit surface being a sphere. We assume there is a bounding box surrounding the sphere. Then we can use one of the faces of the box as the proxy.

Let $\mathcal{F}$ be one of the faces of the bounding box of the sphere $\mathcal{B}$. We consider the following integral defined over $\mathcal{B}$ and try to change the domain to $\mathcal{F}$.

$$\int_{\mathcal{B}} h(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \tag{24}$$

We next define the transformations needed. For any point $\boldsymbol{q} \in \mathcal{F}$, we can cast a ray $r := \{\boldsymbol{q} + t\boldsymbol{n}_{\mathcal{F}}(\boldsymbol{q}) \,|\, t \in \mathbb{R}^+\}$ from $\boldsymbol{q}$ toward $\mathcal{B}$. This ray will intersect with $\mathcal{B}$ two times. Let $\mathcal{T}_1 : \mathcal{F} \to \mathcal{B}$ be the transformation mapping $\boldsymbol{q} \in \mathcal{F}$ to the first intersection and $\mathcal{T}_2$ the second. $\mathcal{T}_1$ and $\mathcal{T}_2$ cut $\mathcal{B}$ into two disjoint parts and so do the integral:

$$\int_{\mathcal{T}_1(\mathcal{F})} h(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} + \int_{\mathcal{T}_2(\mathcal{F})} h(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \tag{25}$$

Next we change the domains to $\mathcal{F}$:

$$\begin{aligned} &\int_{\mathcal{F}} h(\mathcal{T}_1(\boldsymbol{q}))J_{\mathcal{T}_1}(\boldsymbol{q})\,\mathrm{d}\boldsymbol{q} + \int_{\mathcal{F}} h(\mathcal{T}_2(\boldsymbol{q}))J_{\mathcal{T}_2}(\boldsymbol{q})\,\mathrm{d}\boldsymbol{q} \\ &= \int_{\mathcal{F}} \sum_{i=1}^{2} h(\mathcal{T}_i(\boldsymbol{q}))J_{\mathcal{T}_i}(\boldsymbol{q})\,\mathrm{d}\boldsymbol{q} \end{aligned} \tag{26}$$

where $J_{\mathcal{T}_i}$ are the Jacobian determinants of $\mathcal{T}_i$.

In general, $\mathcal{B}$ could be any surface, meaning there could be *more than two intersections*, and each point in $\mathcal{F}$ could have a *different number of intersections*. For the former problem, we simply continue the summation on. The latter means $\mathcal{T}_i$ might not be defined for some points in $\mathcal{F}$ since the i-th intersection might not exist. We let those points have zero contribution as follows:

$$g_i(\boldsymbol{q}) := \begin{cases} h(\mathcal{T}_i(\boldsymbol{q}))J_{\mathcal{T}_i}(\boldsymbol{q}), & \mathcal{T}_i(\boldsymbol{q}) \text{ defined} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

Letting N be the max number of intersections, we can change the domain of Eq. (24) from the implicit surface $\mathcal{B}$ to the face $\mathcal{F}$ of the bounding box as follows:

$$\sum_{i=1}^{N} \int_{\mathcal{F}} g_i(\boldsymbol{q})\,\mathrm{d}\boldsymbol{q}. \tag{28}$$

Then we can estimate Eq. (28) by directly sampling points from its domain $\mathcal{F}$.

ALGORITHM 1: Our primary-edge sampling routine as in Figure 5-(b)

Sample_Primary_Edge()
Input: Face $\mathcal{F}$ of bounding box and edge $\mathcal{E}$ of $\mathcal{F}$, and max number of steps $i_{\max}$ and step size ϵ
Output: $\boldsymbol{p} \in \mathcal{S}$
begin
  /* Sample a point from the proxy */
  Sample a point $\boldsymbol{q} \in \mathcal{E}$;
  /* ψ_{prev} stores ψ's value at the previous step and is initialized to zero */
  $\psi_{\text{prev}} \leftarrow 0$;
  /* March along the normal of $\mathcal{E}$ over $\mathcal{F}$ */
  for $i = 1, 2, \ldots, i_{\max}$ **do**
    /* Intersect the ray starting from $\boldsymbol{q}$ along the normal $\boldsymbol{n}_{\mathcal{F}}$ of face $\mathcal{F}$ toward implicit surface ϕ */
    $\boldsymbol{p} \leftarrow$ ray_intersect$(\boldsymbol{q}, \boldsymbol{n}_{\mathcal{F}})$;
    if $\boldsymbol{p}$ == NOT_HIT **then**
      $\psi_{\text{prev}} \leftarrow 0$;
    else
      /* Evaluate ψ at the current step */
      $\psi_i \leftarrow \psi(\boldsymbol{p})$;
      /* If sign of ψ at this step is different from that at the last step */
      if $\psi_{\text{prev}} \cdot \psi_i < 0$ **then**
        /* We just intersected ψ, meaning $\boldsymbol{p}$ intersects both ϕ and ψ */
        return $\boldsymbol{p}$;
      end
      $\psi_{\text{prev}} \leftarrow \psi_i$;
    end
    /* Step ahead along the normal $\boldsymbol{n}_{\mathcal{E}}$ of edge $\mathcal{E}$ */
    $\boldsymbol{q} \leftarrow \boldsymbol{q} + \epsilon \cdot \boldsymbol{n}_{\mathcal{E}}$
  end
  return NULL
end

5.2.2 *Next Event Estimation.* After sampling a surface point, we could simply uniformly sample a tangent direction. But we can do it more efficiently using next-event estimation. We do it by intersecting the tangent plane with light and then sampling points from the intersection as in Figure 7-(b). For environment maps, we

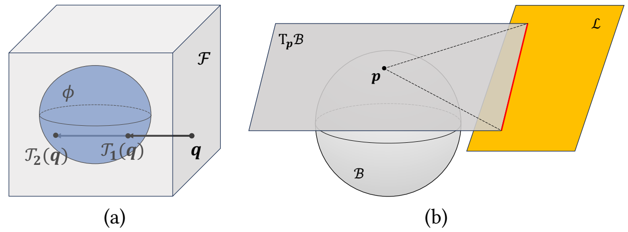


Figure 7: (a) We sample points from the implicit surface ϕ by sampling a point q from the proxy $\mathcal{F}$ and project q onto the implicit surface ϕ. We then take all the intersections, $\mathcal{T}_1(q)$ and $\mathcal{T}_2(q)$ in this example, as the sample points from the implicit surface ϕ. (b) We intersect the tangent plane $\mathrm{T}_p\mathcal{B}$ at p with light $\mathcal{L}$ and only sample from tangent directions intersecting the light $\mathcal{L}$

pre-compute some plane-sphere intersections with uniformly distributed normals determining the plane orientation. We then build a 1D distribution conditioned on the normal direction off of those intersections. This way, we do not have to build distributions on the fly. We assume the number of lights is one in our experiments and leave efficient plane intersection with a large number of polygons to future work.

6 EXPERIMENTS

To assess the effectiveness of our method, we conduct experiments on several test scenes.

6.1 Settings

We implement our Monte Carlo estimator with both *unidirectional* and *bidirectional* path sampling based on the `cuda_ad_rgb` backend of Mitsuba 3 [Jakob et al. 2022]. We conduct all our experiments on an NVIDIA RTX 4090 GPU.

We represent an implicit surface as a grid-based SDF with tricubic interpolation as in diffsdf [Vicini et al. 2022] so that the surface normal is continuous.

Optimization setup. We use the Adam optimizer [Kingma and Ba 2014] and the L2 loss in all our inverse-rendering experiments. The number of views is eight, and the batch size is two. The runtime of one optimization is typically 30 to 90 minutes which usually uses 64 to 256 epochs.

6.2 Differentiable Rendering

We validate our estimators in Figure 8 by comparing derivative estimated by our estimators and that by finite difference using a large number of samples. In the top row, we differentiate the image w.r.t. the horizontal translation of the shape. In the bottom row, we differentiate the image w.r.t. the vertical-axis rotation of the shape. The area light near the ceiling is flipped upside down, and the scene is thus dominated by indirect illumination.

In Figure 9, we show that our bidirectional estimator performs better than our unidirectional one with hard-to-sample lights and specular materials.

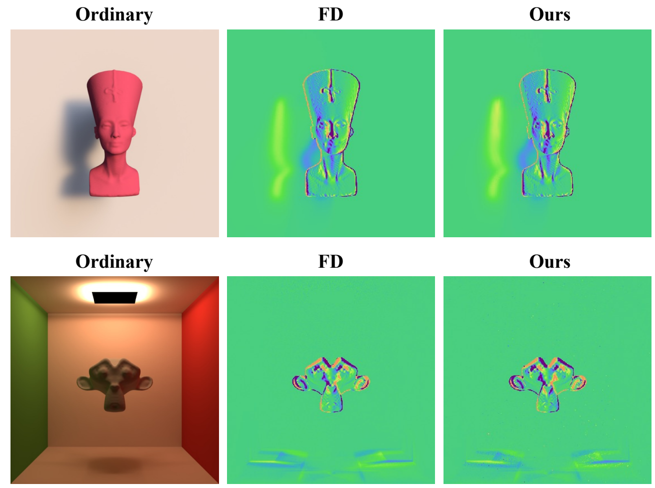


Figure 8: We validate our method by comparing derivative images estimated by our method to that by finite difference (FD) with a large number of samples. Our NEFERTITI and SUZANNE results are computed using our *unidirectional* and *bidirectional* estimators, respectively. Our results closely match the FD reference and demonstrate the correctness of our implementation.

6.3 Inverse Rendering

We quantitatively compare our optimized shape with diffsdf's using Chamfer distance (between target meshes and those extracted from optimized implicit surfaces).

In Figure 12, the light is blocked by the ceiling with only a small hole letting light in. And We only observe the shadow of the shape, so the gradient only consists of the secondary boundary integral. Since light is occluded by the ceiling except for a small region, it is very hard to sample light in this configuration. Our guiding grid helps us to find the un-occluded region of the light and makes our derivative estimates less noisy.

In Figure 13, the scene is lit by a small light flipped upside down. The light is hard to sample in this scene and so our bidirectional estimator is more efficient than the other unidirectional estimators. Our bidirectional estimator's derivative image is cleaner, and the optimized shape is closer to the target and smoother than the other estimators.

In Figure 10 and Figure 11, our estimator computes cleaner gradient for all the three components, leading to much better results than diffsdf.

7 DISCUSSION AND CONCLUSION

Limitations. As is the case with all PSDR methods, sampling boundary paths efficiently is nontrivial and typically requires some form of guiding. Our current implementation uses a simple regular-grid-based scheme (similar to [Zhang et al. 2020]) which is known to have difficulties handling reflections off of highly glossy surfaces. Fortunately, we expect using concurrent projective sampling [Zhang et al. 2023] to greatly mitigate this problem.

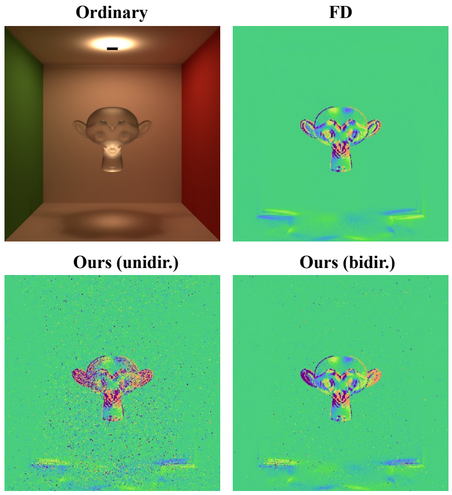


Figure 9: Differentiable rendering comparison (SUZANNE) between our *bidirectional* and *unidirectional* estimator. Scenes where lights are difficult to sample and with specular materials need to be handled by bidirectional estimators.

Future work. How to efficiently guide the new secondary boundary integral for implicit surfaces is something one can explore. One thing that is hard about it is that for one sample on the proxy shape, there are multiple corresponding surface points. We only use a single guiding grid, and that means the tangent direction is fixed over those possibly multiple surface points. We only did experiments on a grid-based SDF while neural SDF is a more popular representation. Theoretically, our theory supports arbitrary continuous level-set functions —including neural SDFs. In practice, we build our system based on diffsdf's implementation (a custom branch of Mitsuba) that uses grid-based SDFs. Integrating simple neural SDFs (e.g., those using small MLPs) into the system should be possible, but supporting more complex neural representations using Dr.Jit efficiently can be challenging (at the system level).

Bias. Theoretically, our formulation is unbiased. In practice, small amounts of bias can be introduced by the ray-surface intersection and the sampling of silhouette points (needed by the primary boundary paths) as follows. Computing the intersection between a ray and an implicit surface (using, for example, spherical tracing) is normally approximated. In other words, the computed intersection is normally very close to the surface but may not be exactly on it. We note that existing methods like diffsdf also have this type of bias. When finding the silhouette points, our method assumes at most one intersection for each sample point on the proxy curve—which may be violated when the AABBs are not small enough. Fortunately, we found the bias to have little impact on the reconstruction quality in all our experiments.

Conclusion. In this paper, we generalized path-space differentiable rendering to support implicit surfaces. Specifically, we demonstrated how material-form parameterization can be realized without requiring a global parameterization of the reference surface. Also, we derived multi-directional form of boundary path integrals which are known to be essential for efficient estimation of these integrals. Based on our generalized formulation, we introduced new Monte Carlo estimators—including bidirectional ones—capable of rendering implicit surfaces with complex light transport effects.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive suggestions. We also thank Heng Guo for his advice and insights. This work was partially supported by JSPS KAKENHI grant JP23H05491 and NSF grant 1900927.

REFERENCES

Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable Rendering of Neural SDFs through Reparameterization. In *ACM SIGGRAPH Asia 2022 Conference Proceedings* (Daegu, Republic of Korea) *(SIGGRAPH Asia '22)*. Association for Computing Machinery, New York, NY, USA, Article 22, 9 pages. https://doi.org/10.1145/3550469.3555397

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

Alan H Barr. 1986. Ray tracing deformed surfaces. *ACM SIGGRAPH Computer Graphics* 20, 4 (1986), 287–296.

Robert A Drebin, Loren Carpenter, and Pat Hanrahan. 1988. Volume rendering. *ACM Siggraph Computer Graphics* 22, 4 (1988), 65–74.

Jorge Flórez, Mateu Sbert, Miguel A Sainz, and Josep Vehí. 2008. Efficient ray tracing using interval analysis. In *Parallel Processing and Applied Mathematics: 7th International Conference, PPAM 2007, Gdansk, Poland, September 9-12, 2007 Revised Selected Papers 7*. Springer, 1351–1360.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Marc Levoy. 1988. Display of surfaces from volume data. *IEEE Computer graphics and Applications* 8, 3 (1988), 29–37.

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), 222:1–222:11.

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* 38, 6 (2019), 228:1–228:14.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 146:1–146:15.

Jos Stam and Ryan Schmidt. 2011. On the Velocity of an Implicit Surface. *ACM Trans. Graph.* 30, 3, Article 21 (may 2011), 7 pages. https://doi.org/10.1145/1966394.1966400

Jorge Stolfi and Luiz Henrique De Figueiredo. 1997. Self-validated numerical methods and applications. In *Brazilian mathematics colloquium monograph*. IMPA ˆ eRio de Janeiro Rio de Janeiro, 1–116.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Vol. 1610. Stanford University PhD thesis.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths Using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4, Article 108 (2021), 108:1–108:14 pages.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Trans. Graph.* 41, 4 (2022), 125:1–125:18.

Andrew P Witkin and Paul S Heckbert. 1994. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 269–277.

Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* 42, 6 (2023), 213:1–213:18.

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), 123:1–123:13.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. *Advances in Neural Information Processing Systems* 33 (2020).

Z. Yu, C. Zhang, O. Maury, C. Hery, Z. Dong, and S. Zhao. 2023. Efficient Path-Space Differentiable Volume Rendering With Respect To Shapes. *Computer Graphics Forum* 42, 4 (2023).

Zihan Yu, Cheng Zhang, Derek Nowrouzezahrai, Zhao Dong, and Shuang Zhao. 2022. Efficient Differentiation of Pixel Reconstruction Filters for Path-Space Differentiable Rendering. *ACM Trans. Graph.* 41, 6 (2022), 191:1–191:16.

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-space differentiable rendering. *ACM Trans. Graph.* 39, 4 (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.* 38, 6 (2019), 227:1–227:16.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021. Path-space differentiable rendering of participating media. *ACM Trans. Graph.* 40, 4 (2021), 76:1–76:15.

Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 42, 6 (Dec. 2023). https://doi.org/10.1145/3618385

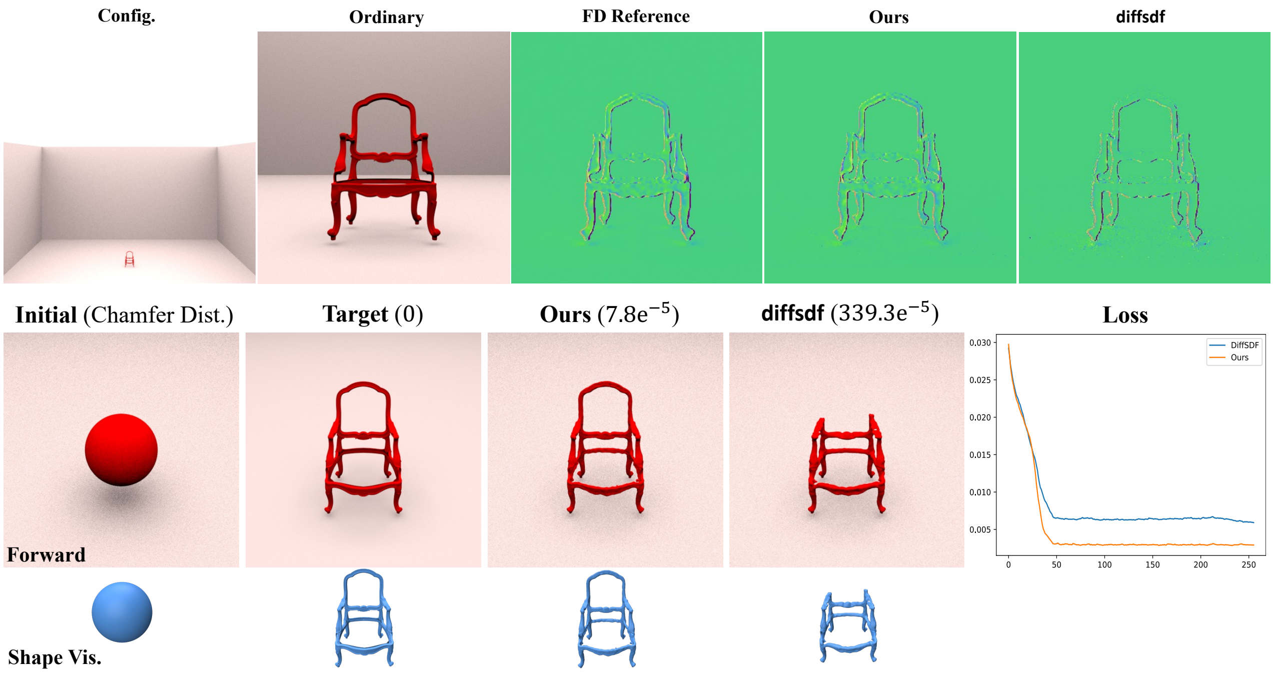


Figure 10: Differentiable and inverse rendering comparison (CHAIR) between our *unidirecitional* estimator and diffsdf [Vicini et al. 2022]. Our estimator can be used in inverse rendering to recover complex topology and thin features such as the chair in this scene.

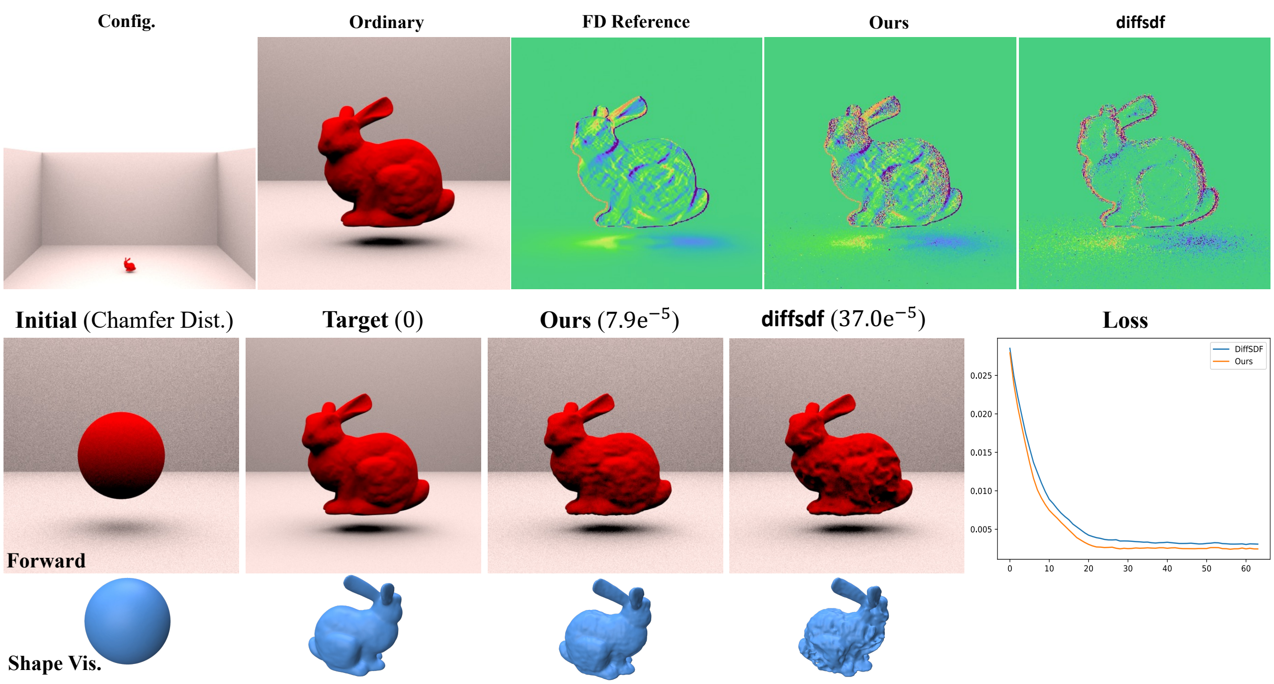


Figure 11: Differentiable and inverse rendering comparison (BUNNY) between our *unidirecitional* estimator and diffsdf [Vicini et al. 2022]. Our estimator estimates all the three components better than diffsdf and leads to the optimized shape being smoother.

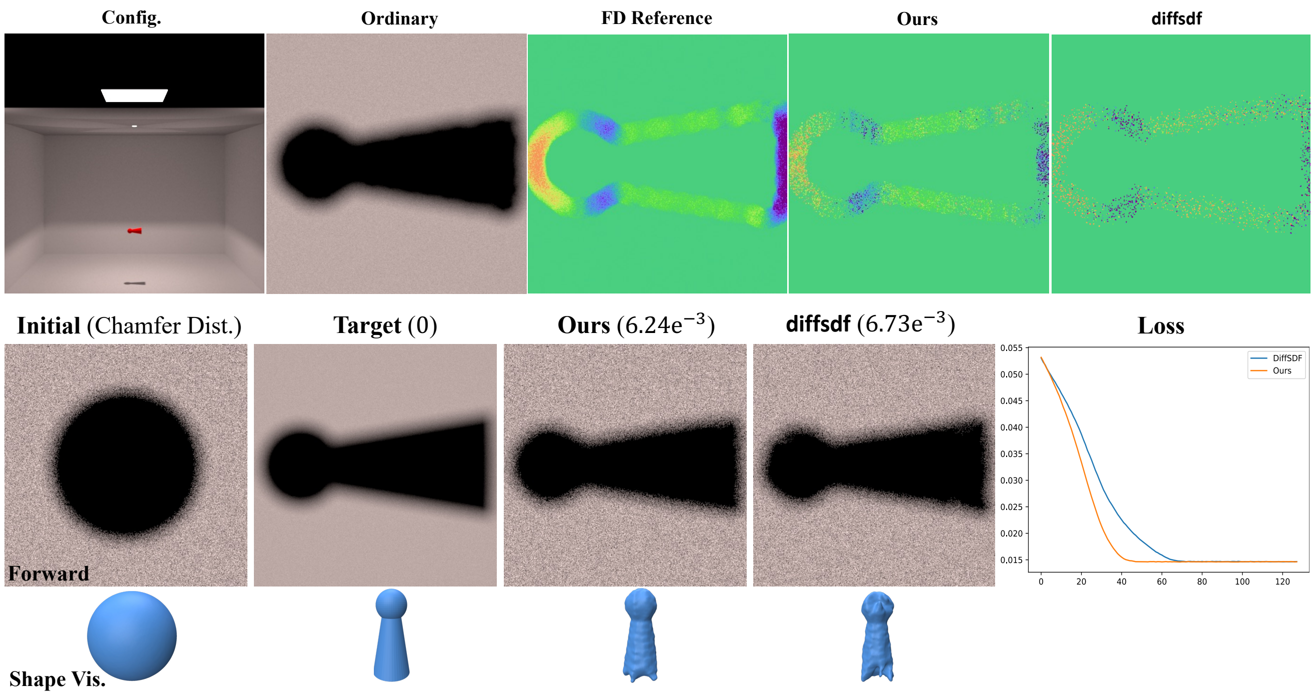


Figure 12: Differentiable and inverse rendering comparison (SHADOW) between our *unidirecitional* estimator and diffsdf [Vicini et al. 2022]. We compare the secondary boundary integral of our estimator and diffsdf's in this scene where we only observe the shadow of the shape. We achieve less variance in the secondary boundary integral by explicit sampling this component and guiding.

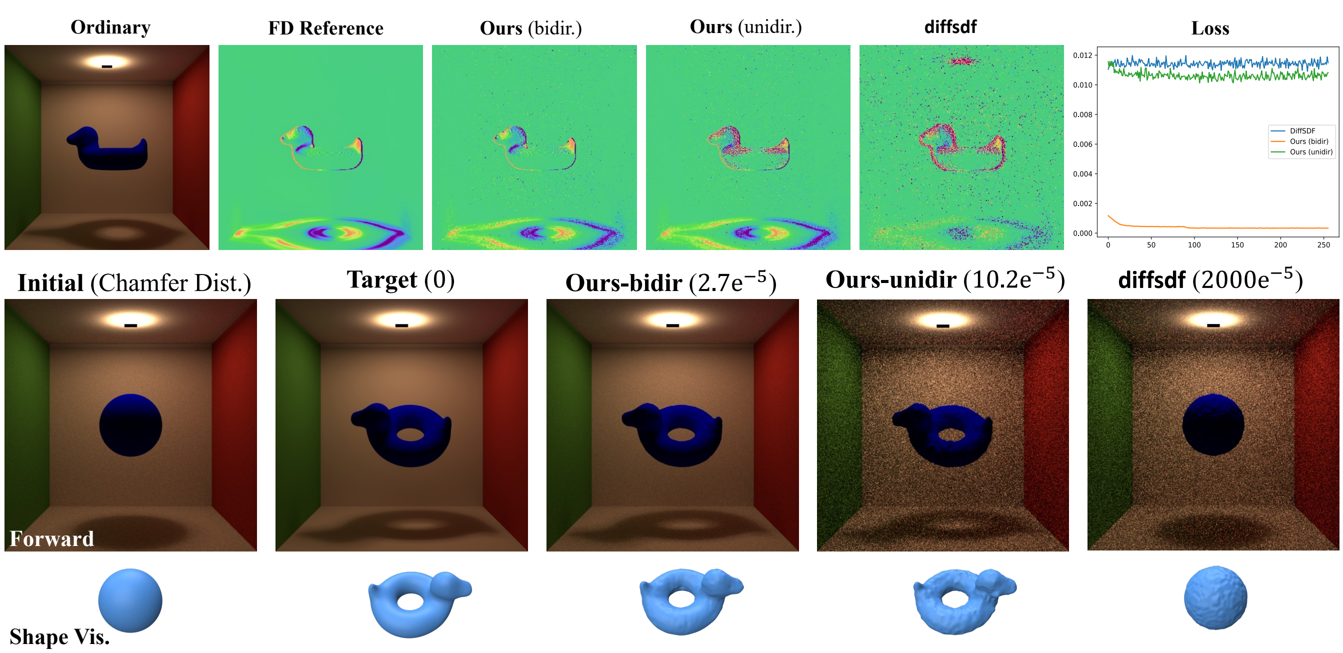


Figure 13: Differentiable and inverse rendering comparison (BOB) between our *bidirectional* estimator, *unidirectional* estimator and diffsdf [Vicini et al. 2022]. With scenes where lights are difficult to sample, our bidirectional estimator can achieve less variance than other unidirectional estimators.