Differentiable Rendering of Neural SDFs through Reparameterization

Sai Praveen
Bangaru
MIT CSAIL
USA
sbangaru@mit.edu

Michaël Gharbi
Adobe Research
USA
mgharbi@adobe.com

Tzu-Mao Li
UC San Diego
USA
tzli@ucsd.edu

Fujun Luan
Adobe Research
USA
fluan@adobe.com

Kalyan Sunkavalli
Adobe Research
USA
sunkaval@adobe.com

Miloš Hašan
Adobe Research
USA
mihasan@adobe.com

Sai Bi
Adobe Research
USA
sbi@adobe.com

Zexiang Xu
Adobe Research
USA
zexu@adobe.com

Gilbert Bernstein
MIT CSAIL &
UC Berkeley
USA
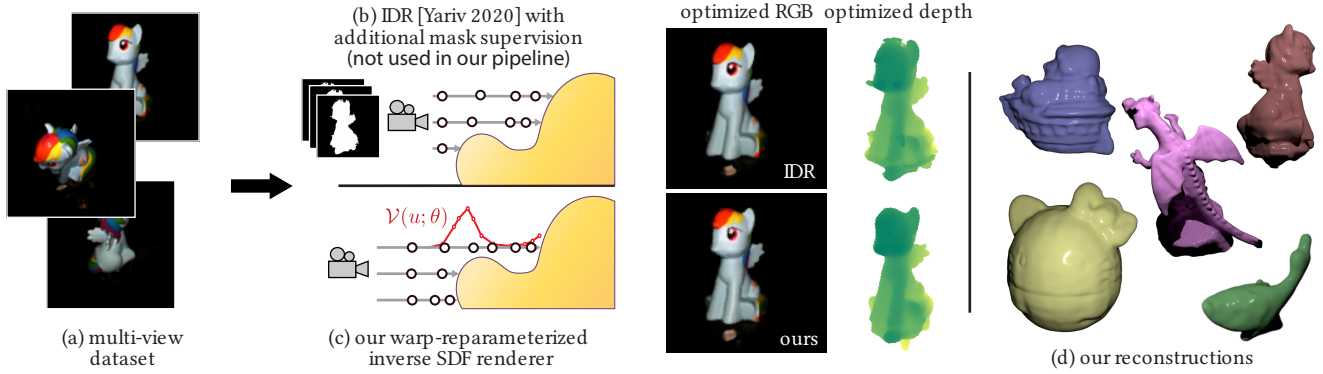gilbo@berkeley.edu

Frédo Durand
MIT CSAIL
USA
fredo@mit.edu

Figure 1: We propose a novel method to correctly differentiate a neural SDF renderer by reparameterizing the pixel integral. Direct application of automatic differentiation to the renderer fails because of discontinuities like silhouette boundaries. In this work we show that, by carefully designing a *discontinuity-aware* warp function $\mathcal{V}(u; \theta)$ to reparameterize the pixel domain, we can remove these discontinuities, and the reparameterized integral is amenable to automatic differentiation. We demonstrate the benefits of our method on inverse rendering problems. Starting from a multiview dataset of real photos (a), our reparameterized renderer (c) can optimize a neural SDF that closely matches the input data, and generalizes to novel views. Our renderer matches or outperforms prior SDF renderers [Yariv et al. 2020] (b), while doing away with their need for additional geometric supervision in the form of per-view masks, which can be unreliable for real-world data. We show additional surface reconstructions obtained with our inverse renderer in (d).

ABSTRACT

We present a method to automatically compute *correct* gradients with respect to geometric scene parameters in neural SDF renderers. Recent physically-based differentiable rendering techniques for meshes have used *edge-sampling* to handle discontinuities, particularly at object silhouettes, but SDFs do not have a simple parametric form amenable to sampling. Instead, our approach builds on *area-sampling* techniques and develops a continuous warping function for SDFs to account for these discontinuities. Our method leverages the distance to surface encoded in an SDF and uses quadrature on sphere tracer points to compute this warping function. We further show that this can be done by subsampling the points to make the method tractable for neural SDFs. Our differentiable renderer can be used to optimize neural shapes from multi-view images and produces comparable 3D reconstructions to recent SDF-based inverse rendering methods, without the need for 2D segmentation masks to guide the geometry optimization and no volumetric approximations to the geometry.

CCS CONCEPTS

• **Mathematics of computing** → **Differential calculus**; *Integral calculus*; **Calculus**; • **Computing methodologies** → *Modeling and simulation*; **Computer vision**; **Computer vision representations**; **Rendering**; **Ray tracing**.

KEYWORDS

differentiable rendering, signed distance functions, neural SDFs

1 INTRODUCTION

Differentiable rendering algorithms have become crucial tools in solving challenging inverse problems [Zhao et al. 2020], thanks to their ability to compute the derivatives of images with respect to arbitrary scene parameters. Naive differentiation of rendering algorithms does not handle discontinuities caused by visibility changes and object boundaries correctly. Previous work has observed that the discontinuities can be handled by properly handling the Dirac delta signals, and derived algorithms for explicit geometry representations like triangle meshes [Li et al. 2018; Zhang et al. 2020].

On the other hand, implicit representations like signed distance fields (SDFs) are appealing since they do not require the initialized geometry to have the right topology. Recent work has demonstrated the use of SDFs—usually parameterized using multi-layer perceptron networks—for the task of reconstructing shape and surface reflectance from images. However, these methods either require additional geometric supervision such as segmentation masks [Yariv et al. 2020; Zhang et al. 2021b] or make approximations to the geometry using volumetric models [Oechsle et al. 2021; Yariv et al. 2021] that limit their applicability.

In this paper, we derive an algorithm to automatically compute *correct* gradients with respect to geometric scene parameters in neural SDF renderers. Previous methods that rely on silhouette sampling are not directly applicable to SDFs since direct sampling of boundaries of implicit functions is challenging. Instead, we build on the *reparameterization* approaches [Bangaru et al. 2020; Loubet et al. 2019], which removes discontinuities through reparameterization while preserving the integral values. These methods do not require explicit sampling along discontinuities. Previous reparameterization methods focused on triangle meshes, and require new derivation for reparameterizing SDF rendering.

Specifically, we construct a silhouette-aware reparameterization similar to that of Loubet et al. [2019], but following the equivalent unbiased warp definition that Bangaru et al. [2020] used to produce correct gradients for triangle-meshes. We leverage the fact that SDFs naturally encode the distance to the surface, and develop a practical algorithm that uses a quadrature on sphere tracing [Hart 1996] samples to construct a reparameterization that removes the discontinuities. We further show that this can be computed using only a subset of sphere tracing samples, reducing the computational burden of the backward pass for bulky neural SDFs.

Our algorithm produces correct geometry gradients for SDFs. It does away with the segmentation masks and depth guidance required by previous techniques [Yariv et al. 2020], without making a volumetric approximation to the geometry [Oechsle et al. 2021; Yariv et al. 2021]. We show that our differentiable renderer can be used to optimize neural shapes from multi-view images, with no additional information beyond the RGB data and the corresponding camera parameters. Our focus is on occlusion discontinuities, so the rest of the paper assumes a differentiable shading model.

2 RELATED WORK

We focus on work that recovers the latent 3D scene from images through differentiable rendering. We categorize them by the type of scene representation.

Meshes. To account for discontinuities, earlier work focused on approximating the derivatives of mesh rendering by smoothing the geometry [de La Gorce et al. 2011; Kato et al. 2018; Liu et al. 2019; Loper and Black 2014; Rhodin et al. 2015]. Alternatively, some work derived correct analytical derivatives under simplified assumptions [Arvo 1994; Zhou et al. 2021]. Li et al. [2018] noticed that the differentiation of discontinuities caused by the visibility and geometric boundaries lead to Dirac delta signals, and can be integrated by the pixel antialiasing integral or the rendering equation. They proposed an *edge sampling* algorithm to explicitly sample the Dirac delta on triangle mesh silhouettes. Importance sampling the silhouettes can be difficult, therefore Loubet et al. [2019] and Bangaru et al. [2020] later proposed to convert the silhouette integral into an area integral. Loubet et al. formulated the conversion using a reparameterization, and derived an approximated reparametrization to remove discontinuities. Bangaru et al. built on Loubet et al.'s work and derived an unbiased estimator by showing the equivalence between the reparameterization and divergence theorem. On the other hand, Zhang et al. [2020] showed that directly sampling silhouette in path-space [Veach 1998] can also be done efficiently. Directly sampling the silhouette for SDFs is difficult. Our work extends the reparameterization approach to handle SDFs, including approximate SDFs defined by neural networks.

Level sets and signed distance fields. A level set defines a surface using the roots of a 3D implicit function. A signed distance field is a specific kind of level set where the implicit function defines the distance of a 3D point to the surfaces, where the sign is negative when the point is inside the object. SDFs can be represented using polynomials [Blinn 1982], voxels [Izadi et al. 2011], or neural networks [Park et al. 2019]. Differentiable rendering for SDFs has been discussed in computer vision and used for 3D surface reconstruction [Jiang et al. 2020; Kellnhofer et al. 2021; Niemeyer et al. 2020; Yariv et al. 2020; Zhang et al. 2021b], but current methods all ignore the discontinuities when differentiating, and require 2D object masks to converge. An alternative way to render the signed distance field is to convert it to another format such as a thin participating medium [Oechsle et al. 2021; Wang et al. 2021; Yariv et al. 2021], a mesh [Remelli et al. 2020], or a point cloud [Cole et al. 2021]. These methods all introduce approximation. Instead, we focus on deriving accurate gradients without approximation. Our work is closely related to the concurrent work by Vicini et al. [2022]. They too build on the warped-area sampling formulation from Bangaru et al. [2020], and arrive at a similar harmonic weighting scheme applied to sphere tracer points. The main difference is that we apply our method to the neural SDF and radiance model proposed by Yariv et al. [2020] to remove their mask requirement in a principled
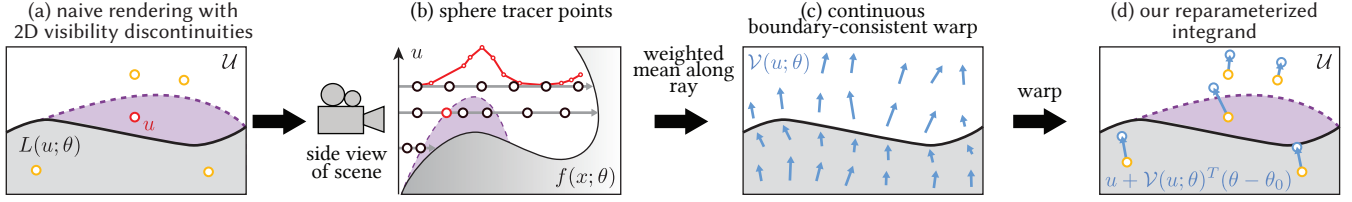
Figure 2: OVERVIEW. A standard SDF rendering pipeline is generally discontinuous, which means there are points u where the rendering function $L(u;\theta)$ is not differentiable in θ, highlighted in red (a). Our method uses intermediate points from a sphere tracer (b) applied to an SDF f, to compute a warp function $\mathcal{V}$ (c). Using this warp, we reparameterize the integration domain to avoid discontinuities (d), which allows us to compute correct gradients of the rendering equation. The key to achieving this is to design the warp $\mathcal{V}$ so it is continuous in u everywhere, and satisfies some consistency criterion on the geometric boundaries.

manner. Vicini et al. [2022], on the other hand, experiment with voxelized SDFs and analytical BRDFs. The higher memory requirement of neural SDFs motivated our top-k version of the harmonic weights. We also focus on primary visibility since our main goal is to lift the mask requirement in neural SDF rendering.

Volumes. A scene can also be represented as participating media instead of solid surfaces. Gkioulekas et al. [2013] pioneered the use of differentiable volume rendering for inverse problems. Zhang et al. [2019; 2021c] tackled discontinuities at volumetric boundaries. Recently, there has been a surging interest in using volumetric representations—parameterized either as discretized grids or neural networks—for view synthesis [Liu et al. 2020; Lombardi et al. 2019; Mildenhall et al. 2020; Xie et al. 2022]. These volumetric representations allow for a trivially differentiable rendering model and can achieve high-quality novel view synthesis and appearance acquisition [Bi et al. 2020a,b]. However, it is still a challenge to extract high-quality surface geometry from these methods, and while the trade-offs between surface and volume representations is an interesting research topic, we focus on surface representations.

Light transport. In addition to handling discontinuities, recent work also studies the reduction of variance and memory consumption for Monte Carlo rendering [Nimier-David et al. 2020; Vicini et al. 2021; Zeltner et al. 2021; Zhang et al. 2021a]. Earlier rendering work used derivatives for forward rendering [Li et al. 2015; Luan et al. 2020; Ramamoorthi et al. 2007; Ward and Heckbert 1992]. Our work is largely orthogonal to these.

3 METHOD

Our method computes the correct gradient of a rendering function (i.e., the pixel integral of the radiance function on the camera image plane) with respect to geometric parameters, in the presence of primary visibility discontinuities, for scenes where the geometry is represented by a signed distance field f, parameterized by θ (e.g., the weights of neural network). Our approach builds on Bangaru et al. [2020]. We show how to extend their warp function to SDFs in order to reparameterize an intractable boundary integral. We summarize the necessary background in § 3.1. We then derive a warp function for SDFs that is continuous and boundary consistent (§ 3.2) as an integral along camera rays, and show how to compute it via quadrature using sphere tracer points (§ 3.3). In Section 3.4, we finally give an unbiased approximation for this warp that is

tractable for use with neural SDFs. Section 3.5 provides details on how to use our approach to solve inverse rendering problems.

3.1 Background: boundary-aware warping

Without loss of generality, assume a box pixel filter, so that $\mathcal{U} \subset \mathbb{R}^2$ is the image plane region corresponding to the pixel of interest. Let $L(u;\theta)$ denote the radiance along the ray from $u \in \mathcal{U}$, a point on the image plane, and denote $\theta \in \mathbb{R}^N$ the vector of geometric scene parameters (e.g. neural network weights). In matrix expressions below, we will assume vector quantities (u, x, θ) to be row vectors, and gradients with respect to θ to be column vectors.

We aim to compute the gradient of the rendering integral I with respect to parameters θ:

$$\partial_\theta I = \frac{\partial}{\partial \theta} \int_{\mathcal{U}} L(u;\theta)du. \tag{1}$$

Primary visibility discontinuities make the radiance function non-differentiable along occlusion boundaries (Fig. 3). Denoting $\mathcal{U}_{\text{sil}}(\theta) \subset \mathcal{U}$ the set of object silhouettes, for a point $u_{\text{sil}} \in \mathcal{U}_{\text{sil}}$, the radiance $L(u_{\text{sil}};\theta)$ is discontinuous in θ. This makes naive automatic differentiation methods applied to the Monte Carlo sampling of I produce incorrect gradients since they ignore the Dirac delta that arises from the differentiation.

Li et al.[2019; 2018] and Zhang et al. [2019] showed that Eq. (1) can be split into two terms: an interior integral, for contributions away from discontinuities and a boundary integral, along discontinuities:

$$\partial_\theta I = \int_{\mathcal{U}} \frac{\partial}{\partial \theta} L(u;\theta)du + I_{\text{sil}}. \tag{2}$$

The second integral I_{sil} is harder to compute because sampling the boundary is difficult. This is particularly true for SDFs whose surface boundaries admit no easy parametric form. We will not cover boundary sampling in detail, since we will not use it; instead, we will use a result from Bangaru et al. [2020], who showed, using the divergence theorem, that this boundary term can be turned into an integral over the interior $\mathcal{U} \setminus \mathcal{U}_{\text{sil}}(\theta)$, which is easier to sample:

$$I_{\text{sil}} = \int_{\mathcal{U} \setminus \mathcal{U}_{\text{sil}}(\theta)} \nabla_u \cdot (L(u;\theta)\mathcal{V}(u;\theta))\, du. \tag{3}$$

Here $\nabla_u\cdot$ is the divergence operator, and $\mathcal{V}(u;\theta) \in \mathbb{R}^{N \times 2}$ is a warping function required to satisfy two properties:

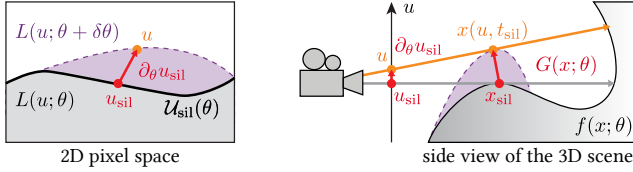(1) **continuity**: $\mathcal{V}(\cdot;\theta)$ is continuous on $\mathcal{U}$, and

2D pixel space   side view of the 3D scene

Figure 3: As geometric parameters θ vary, visibility creates discontinuities in the rendering function $L(u;\theta)$ which traditional automatic differentiation cannot handle (left). These discontinuities correspond to continuous changes in our SDF representation $f(x;\theta)$ (right). We compute the derivative $\partial_\theta u_{sil}$ of a pixel-space silhouette point w.r.t. the geometry parameters θ by computing the derivative $G(x;\theta)$ of the corresponding 3D scene point x, and projecting it onto the screen space $\mathcal{U}$ through the inverse Jacobian.

(2) **boundary consistency**: $\mathcal{V}$ agrees with the derivative of the discontinuity points when u approaches the discontinuity. That is, $\lim_{u \to u_{sil}} \mathcal{V}(u;\theta) = \partial_\theta u_{sil}$ for $u_{sil} \in \mathcal{U}_{sil}(\theta)$.

Bangaru et al. further show the area integral is equivalent to applying the change of variable [Loubet et al. 2019] $u \mapsto T(u,\theta) = u + (\theta - \theta_0)\mathcal{V}(u;\theta)$ in Eq. (1), where the derivative is computed at θ_0, but $\partial_\theta \theta_0 = 0$. After reparameterization we get:

$$\partial_\theta I = \int_{\mathcal{U}} \partial_\theta \left[L(T(u,\theta),\theta) \left| \det(\partial_u T(u,\theta)) \right| \right] du. \qquad (4)$$

Expanding T and using Eq. (3), one can show that Eq. (4) indeed computes $\partial_\theta I$. Intuitively, the reparameterization T moves each point on the boundary locally at the velocity of their derivatives, essentially removing the discontinuities, while the determinant term accounts for the change of measure.

The main goal of this paper is to derive a suitable form for $\mathcal{V}(u;\theta)$ for SDFs, that can be tractably computed, so that we can evaluate Eq. (4) using Monte Carlo estimation.

Rendering. To render an SDF f and compute $L(u;\theta)$, we need to find the closest intersection point $x(u,t) \in \mathbb{R}^3$ such that $f(x;\theta) = 0$, where t is the distance along the primary ray associated with pixel location u. To find the intersection distance, we use sphere tracing [Hart 1996], which applies a fixed-point iteration to generate a sequence of points $x_n \in \mathcal{T}(u)$, such that $\lim_{n \to \infty} x_n = x$.

3.2 Continuous boundary-consistent warp for SDFs

In this section, we construct an idealized warp function $\mathcal{V}^{int}$ that satisfies the continuity and boundary-consistency conditions of Section 3.1. First, we derive the boundary gradient $\partial_\theta u_{sil}$ with which the warp should agree at silhouette points (§ 3.2.1). We then smoothly extrapolate this gradient using a weighted integral along the primary ray passing through u, to obtain our warp function (§ 3.2.2). We show necessary and sufficient conditions on the weighs to make the warp continuous and boundary-consistent (§ 3.2.3).

3.2.1 Boundary consistency for implicit functions. The boundary consistency condition in Section 3.1, requires that, at a discontinuity point u_{sil} the warp agrees with $\partial_\theta u_{sil}$. The derivation proposed by Bangaru et al. [2020] does not apply directly to implicit surfaces,

so we derive this boundary derivative using the implicit function theorem. Specifically, the derivative of a scene point $x \in \mathbb{R}^3$ on the surface, i.e., $f(x;\theta) = 0$, w.r.t. parameters $\theta \in \mathbb{R}^N$ is given by:

$$G(x;\theta) := \partial_\theta x = -\frac{\partial_\theta f \partial_x f^T}{\|\partial_x f\|^2} \in \mathbb{R}^{N \times 3}. \qquad (5)$$

The above directly follows from the implicit function theorem applied to $f(x;\theta) = 0$. This implicit velocity formulation also appears in previous work [Remelli et al. 2020; Stam and Schmidt 2011; Vicini et al. 2022]. To get the derivative in pixel coordinates $\partial_\theta u = \partial_\theta x \cdot \partial_x u \in \mathbb{R}^{N \times 2}$, we need to project this derivative by the Jacobian $\partial_x u \in \mathbb{R}^{3 \times 2}$, which for a perspective camera can be easily derived by hand. For more generality, we can obtain this Jacobian as the pseudo-inverse † of the forward Jacobian:

$$\partial_x u = (\partial_u x(u,t))^\dagger. \qquad (6)$$

Taken together, the derivative at a silhouette point u_{sil}, with corresponding 3D position $x_{sil} = x(u_{sil}, t_{sil})$, is then:

$$\partial_\theta u_{sil} = G(x_{sil};\theta)\, \partial_x u. \qquad (7)$$

Figure 3 illustrates the geometric configuration.

3.2.2 Extending to a smooth warp $\mathcal{V}^{int}(u;\theta)$ by integration along the ray. Now that we have an expression for the warp at silhouette points, we extend it to all points, by smoothing this term in a consistent manner. Our method leverages the fact that our implicit SDF $f(x;\theta)$ is continuous in 3D space and achieves smoothing by convolving *along the ray* (Fig. 4 (b)). This avoids casting expensive additional rays which are needed by Bangaru et al. [2020], and also propagates gradients to points in free space *near* the boundary points. While investigating the free space gradient is outside the scope of this paper, previous work [Oechsle et al. 2021; Wang et al. 2021] have noted that this can have a stabilizing effect on the optimization of neural SDFs. Note that, while they adapt a volumetric rendering model for better convergence, we do so while computing correct boundary gradients for a surface-based representation.

Our proposed warp function smoothly extends Eq. (7) to non-boundary points as follows:

$$\mathcal{V}^{int}(u;\theta) = \frac{\int_{t=0}^{t_0} w(x(u,t))\, G(x;\theta)\, \partial_x u\, dt}{\int_{t=0}^{t_0} w(x(u,t))\, dt}, \qquad (8)$$

with t_0 the distance to the closest intersection, $t_0 = \infty$ when the ray does not intersect.

3.2.3 Choice of weights. In order to satisfy the boundary consistency criteria, the weights need to asymptotically satisfy the limit:

$$\lim_{u \to u_{sil}} \frac{w(x(u,t))}{\int_{t'=0}^{t_0} w(x(u,t'))\, dt'} = \delta(t - t_{sil}), \qquad (9)$$

where δ is the Dirac delta operator.

From Eq. (9), we see that our weights have to depend on some notion of distance to the silhouette. For an implicit function f that is at least C_1 continuous, the following constraints implicitly characterize the silhouette points [Gargallo et al. 2007]:

$$\begin{aligned} f(x(u,t);\theta) &= 0, \\ \partial_x f(x(u,t);\theta)^T \partial_t x(u,t) &= 0. \end{aligned} \qquad (10)$$
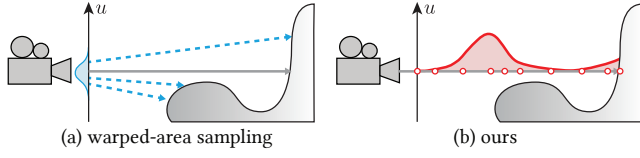
Figure 4: (a) Warped-area sampling uses additional random rays around the primal ray and averages $\partial_\theta x(u; t)$ using boundary-aware harmonic weights. (b) Our method instead takes a weighted average along the ray, repurposing the SDF f into weights $S(x)^{-\gamma}$.

The first condition requires the point to be on the surface, and the second condition requires the SDF gradient to be perpendicular to ray direction [Hertzmann 1999]. We can use these equations to build a silhouette characteristic function $\mathcal{S}(x)$, which takes value 0 whenever x is a silhouette point, and is continuous everywhere. Specifically, we define:

$$\mathcal{S}(x) = |f(x; \theta)| + \lambda_d \left| \partial_x f(x; \theta)^T \partial_t x \right|, \quad (11)$$

where $\lambda_d > 0$. This characteristic function is similar to the boundary test function used by Bangaru et al. [2020] for meshes. However, unlike their boundary test, $\mathcal{S}(x)$ is defined everywhere in the SDF's 3D domain, not just the surface points. This allows us to use these weights for our integral along any ray.

Our final harmonic weights are given by:

$$w(x) = \mathcal{S}(x)^{-\gamma}, \gamma > 2. \quad (12)$$

For $\gamma > 2$, our weights satisfy the limit in Eq. (9). Intuitively, this is because the $w(x) \to \delta(x - x_{\text{sil}})$ as $x \to x_{\text{sil}}$. See our supplementary material for a discussion of correctness, and derivation of $\gamma > 2$.

Fig. 5(a) shows our weight distribution along the ray for all u in an 1D example sphere tracer.

3.3 Estimating the warp through its quadrature $\mathcal{V}^q$

We now have a clear form for our warp function that can be used to reparameterize and differentiate the rendering function. Unfortunately, the asymptotical sharpness of our weights required to obtain a valid warp, also makes the integral (8) very difficult to sample. For u close to the silhouette $\mathcal{U}_{\text{sil}}$, the weights become very concentrated near the surface boundary, presenting a tricky integrand if we were to uniformly sample along the ray.

Careful importance sampling of areas near the boundary could remedy this, but there is unfortunately no straightforward way to implement this: the weight distribution depends heavily on the configuration of silhouettes near u, dictated by the SDF.

Our approach foregoes stochastic sampling altogether. We construct a trapezoidal *quadrature* on the series of intermediate points $x_n \in \mathcal{T}(u)$ generated by the sphere tracer, shown in Fig. 5(b). This quadrature estimator for the warp is given by:

$$\mathcal{V}^q(u; \theta) = \frac{\sum_{x_i \in \mathcal{T}(u)} w^q(x_i) \, G(x; \theta) \, \partial_x u}{\sum_{x_i \in \mathcal{T}(u)} w^q(x_i)},$$
$$\text{where } w^q(x_i) = w(x_i) \frac{(t_{i-1} - t_{i+1})}{2}, \quad (13)$$

and t_i is the distance along the ray to sphere tracer point x_i. Assuming the underlying SDF $f(x; \theta)$ is C_1 continuous, the intermediate points of the sphere tracer are continuous at all $u \notin \mathcal{U}_{\text{sil}}$. By composition of continuous functions, $\mathcal{V}^q(\cdot; \theta)$ is also continuous.

Our quadrature warp $\mathcal{V}^q$ satisfies the continuity and boundary consistency condition (§3.1). Since we apply trapezoidal quadrature, $\mathcal{V}^q(u; \theta)$ is in general a biased estimator of integral $\mathcal{V}^{\text{int}}(u; \theta)$. However, the two terms *are* equal in the limit as u approaches the silhouette, i.e., for $u_{\text{sil}} \in \mathcal{U}_{\text{sil}}$, $\lim_{u \to u_{\text{sil}}} \mathcal{V}^q(u; \theta) = \lim_{u \to u_{\text{sil}}} \mathcal{V}^{\text{int}}(u; \theta)$, and since the right-hand side is boundary consistent, so is our quadrature warp $\mathcal{V}^q$. See supplemental for a sketch proof of correctness.

3.4 Top-k subset weighting $\bar{w}_k$ to reduce memory use

For complex SDFs such as a neural network, our quadrature warp $\mathcal{V}^q$ has the caveat that it requires back-propagating through *every* sphere tracer point. Previous work like IDR [Yariv et al. 2020] do not have this issue since their (biased) gradient is only computed at the intersection point, and they exclude other points from the gradient computation. Our approach, on the other hand, uses a weighted sum, so we cannot discard intermediate points.

However, as shown in Fig. 5(b), the vast majority of sphere tracer points have negligible weight, and most of the mass is concentrated close to the *silhouette*. We exploit this by only using the subset of points with the highest weight in our warp estimation. That is, instead of using all of $T(u)$, we can instead use a top-k subset $T_k(u)$. Selecting the top-k weights requires adjusting them to ensure that they remain continuous. For a subset size of k, our weights are[1]:

$$\bar{w}_k(x) = \begin{cases} w^q(x) - \min_{x \in T_k(u)} w^q(x), & \text{if } x \in T_k(u) \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

The weights $\bar{w}_k(x)$ still produce a continuous warp field (see supplemental for a sketch of proof). Intuitively, even though the set of points change as a function of u, whenever this change occurs, the points that swap in or out of the set always have weight 0.

3.5 Inverse Rendering Details

In this section, we briefly discuss some details that make our inverse rendering pipeline tractable.

Implementation. Our method requires 3 *nested* derivative passes to (i) compute normals $\partial_x f$, (ii) compute Jacobian of the transformation $\partial_u T$ and (iii) to compute derivatives of the full pipeline $\partial_\theta [L(T(u, \theta)) |\det (\partial_u T(u, \theta))|]$. We use the Python **JAX** automatic differentiation system [Bradbury et al. 2018], which supports nested forward+backward differentiation. We use forward-mode for (i) and (ii), and reverse-mode for (iii). Note that since (i) and (ii) both involve differentiating w.r.t spatial coordinates, the SDF model must be C_1 continuous in $\mathbf{x}$ (but only C_0 continuous in θ). We enforce C_1 continuity through *softplus* non-linearities instead of using ReLUs

Network architecture. For our inverse rendering results, we use the network architecture shown in Fig. 11 of Yariv et al. [2020]. Since our method is slightly more memory-intensive (even with

---

[1]Note that even though we consider the top k weights, only $k - 1$ weights actually have a non-zero contribution to the θ-derivative.
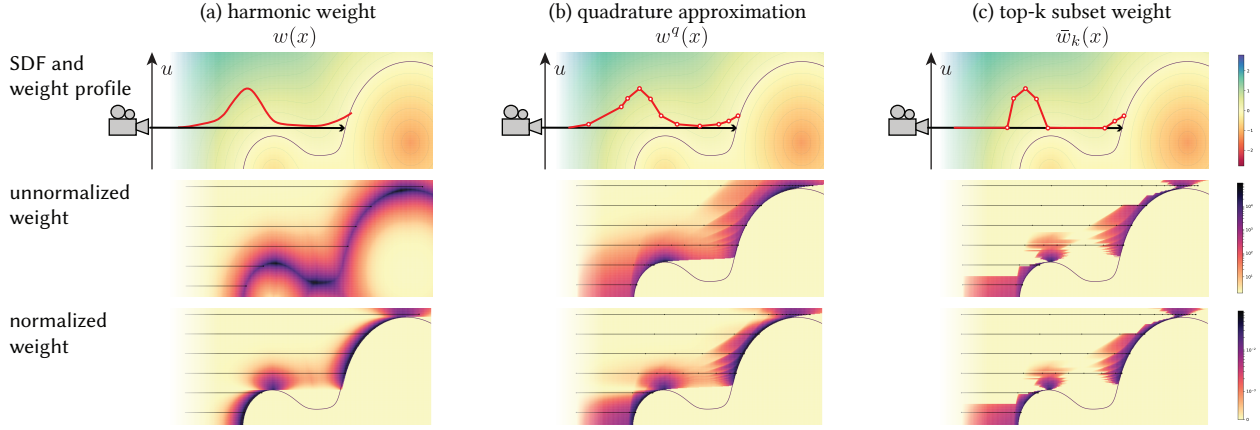
Figure 5: WEIGHT VISUALIZATION. A contour plot of a sample 2D SDF (first row). We use an orthographic camera for illustration, so camera rays are parallel to the horizontal axis. We show our three weighting schemes in unnormalized (second row) and normalized (third row) form. Our proposed harmonic weights (a) for $\gamma = 4.0$, $\lambda_d = 1e - 1$ are well approximated by a trapezoidal quadrature on the sphere tracer points (b). The blank regions with no weight can be excluded from the computation, which leads to our proposed our proposed top-k subset weights (c), for $k = 8$. This reduces both the compute and memory burden of the backward pass. We visualize the weight in a *symlog* plot, values are linear in $[0, 10^1]$ and $[0, 10^{-3}]$ for the unnormalized and normalized weights, respectively.
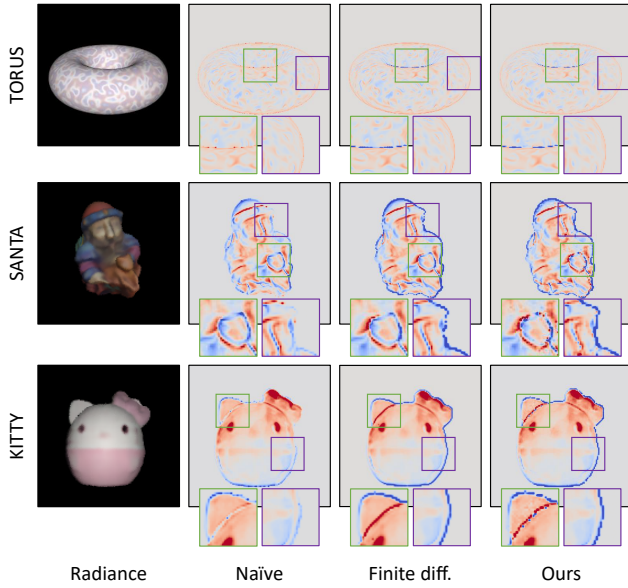

Figure 6: GRADIENT QUALITY. We compare the image gradients computed naïvely without reparameterization [Yariv et al. 2020] and with our method against the "ground truth" gradient computed with finite differences for three scenes. Our method properly handles boundary discontinuities both due to object edges (in purple insets) and self-occlusions (in green insets).

top-k subset weights), we reduce the width of the SDF network to 256 channels per layer. In this architecture, the shading network predicts the final radiance based on the position, viewing direction

and a geometric feature vector. Unlike NeRF [Mildenhall et al. 2020] and its variants, the shading network is only evaluated at *surface* points. We use 6-levels of positional encoding on the input position x to allow the network to reconstruct fine geometry.

Pixel sampling. Similar to Yariv et al. [2020] and other neural methods, we sample a subset of pixels for each iteration since it can be computationally prohibitive to trace the entire image when using a deep neural representation. However, unlike Yariv et al. [2020], which works with a single ray at the center of the pixel, our approach must integrate the spatially-varying warp $\mathcal{V}$ over each pixel. We achieve this by Monte-Carlo sampling within each pixel.

Multi-level optimization. Since we only use a subset of pixels, the likelihood of sampling a pixel with silhouette gradient is fairly low. For unbiased derivatives, only pixels that are partially covered by a surface have a non-zero boundary contribution. This is in contrast to approximate derivatives (e.g., [Liu et al. 2019], [Yariv et al. 2020]) that have a wider spatial footprint. To alleviate this issue, we use a multi-scale pyramid of the target image throughout our optimization.

Initialization. We use the geometric network initialization [Atzmon and Lipman 2020] which approximately produces a spherical SDF. We also initializes the weights of the positional encoding layer to 0 [Yariv et al. 2020]. We found this subtle modification implicitly enforces a coarse-to-fine mechanism that yields significantly better generalization to novel views.

Eikonal constraint. We represent our SDF f using a neural network, which does not necessarily satisfy the distance property. We adopt the Eikonal regularization loss [Gropp et al. 2020] to explicitly enforce this. Since the resulting f is only an approximation of an SDF, we pad our weights with a small ϵ to avoid infinities.
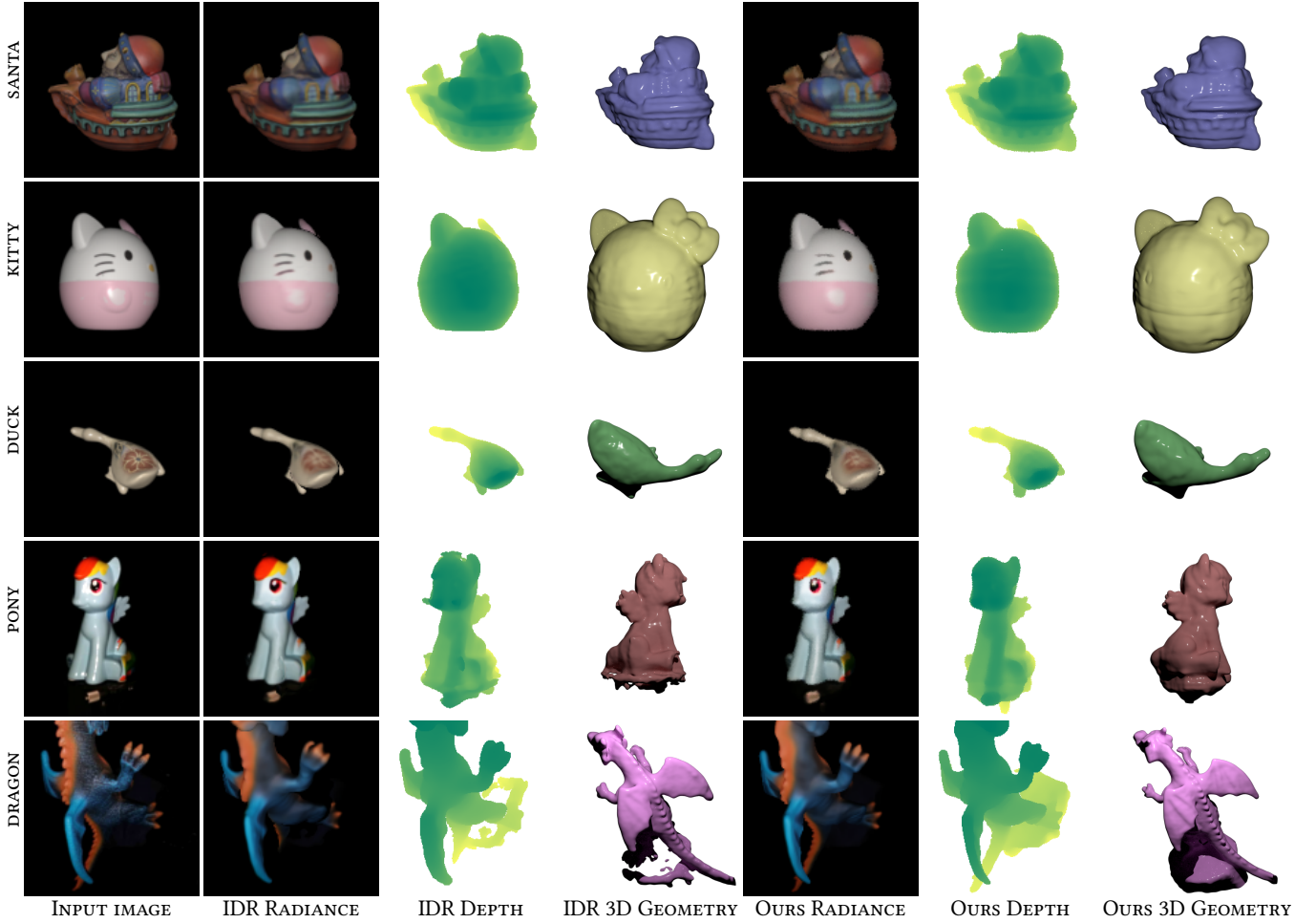
Figure 7: NEURAL SDF RECONSTRUCTION. We compare with IDR [Yariv et al. 2020] on three synthetic scenes (top three rows) and two real captured scene (bottom rows). IDR requires 2D mask supervision, without which it completely diverges. Thanks to our accurate gradient computation, our reconstructions are on par with IDR, without requiring any additional supervision beyond the input images and camera poses. In fact, on the real scenes, our reconstructions (without masks) outperform IDR *with* masks (see the head of the PONY, or the legs, tails and wings of the DRAGON) because of errors in automatic 2D segmentation.

4 RESULTS

4.1 Ground truth gradient comparions

We first evaluate the correctness of our gradient by visualizing gradients on three different scenes (illustrated in Fig. 6). For TORUS—a analytical torus model textured with a diffuse Perlin noise albedo—we visualize the gradients w.r.t the outer radius (distance from the center to the center of the ring). (SANTA and KITTY) are 3D models that we represent as neural SDFs. We take the parameters of the neural SDF from an intermediate iteration during an inverse rendering optimization, and visualize the gradient w.r.t the bias parameter of the last layer output (i.e. the level set perturbation). We also compute the gradient without reparameterization; this is similar to the gradient used in previous SDF-based inverse rendering methods [Yariv et al. 2020]. Note that the interior gradient is largely unaffected by reparameterization, with the gradient at the

silhouettes being the largest benefit of our method, especially at self-occlusions. In the next subsection, we show that this boundary gradient is critical and without it, the inverse rendering diverges.

4.2 Comparisons with IDR

We compare our reconstructions against the SDF-based inverse rendering method of IDR [Yariv et al. 2020]. IDR does not correctly account for the boundary term of gradient of the rendering integral and requires additional supervision, in the form of accurate 2D segmentation masks. We implement IDR in our pipeline to ensure that the only difference is our reparameterization. We use the same network architecture for both methods (See Sec. 3.5 for details), and report results after roughly 25,000 network updates. Note that our method uses more samples (2 in the interior + 4 on each pixel boundary) since we use a Monte-Carlo approach to estimate the warp. IDR only requires one sample, fixed at the center of the pixel.

Figure 7 shows that, on three synthetic scenes (Santa, Kitty and Duck), our method *without* any 2D masks supervision obtains comparable depth and RGB reconstruction as IDR *with* (perfect) mask supervision. We also show reconstructions of a captured real scene (Pony from Bi et al. [2020b]). Here, we provide IDR with 2D masks derived from a COLMAP reconstruction, which has errors. As a result, our reconstruction outperforms IDR on this scene. Table 1 establishes this result using mean reconstructed PSNR.

We also tried to compare with IDR without mask supervision. In most cases, IDR without masks diverges completely because of the lack of gradients from the silhouette. This is similar to the observation made by Oechsle et al. [2021].

Our pipeline takes 10.1s per iteration on a system with 2× RTX 2080Ti, at 24,576 rays per iteration.

Table 1: PSNR Comparison. Our method performs on-par with IDR on all datasets in spite of having no mask supervision.

	IDR (with masks)	Ours (w/o masks)
Santa	34.86 dB	33.59 dB
Kitty	36.96 dB	33.71 dB
Duck	37.30 dB	34.14 dB
Pony	32.06 dB	31.56 dB
Dragon	31.64 dB	32.72 dB

4.3 Ablation study: Subset size

Our top-k weighting scheme reduces the memory footprint of our optimization, but this comes at a cost. The smaller k, the sharper the weight landscape. This can cause high variance that can impede the optimization of fine details. We explore this through an ablation study on the Santa dataset, varying k shown below. We use 36 views for this study, and report results after $20,000$ network updates. Details are resolved for $k \geq 14$, as shown in Fig. 8

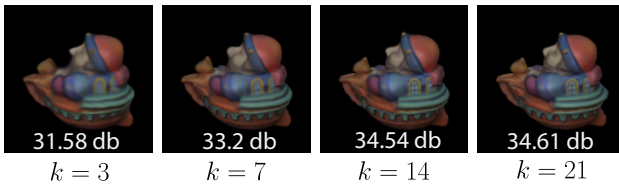| 31.58 db | 33.2 db | 34.54 db | 34.61 db |
| $k = 3$ | $k = 7$ | $k = 14$ | $k = 21$ |

Figure 8: Set Size Ablation. We compare reconstructions (at $25,000$ updates) using different top-k subset sizes from $k = 3$ to $k = 21$, out of a total of 22 sphere tracer steps. We find that beyond $k = 7$, we obtain diminishing returns on reconstruction quality. In this scenario, that translates to roughly 65% fewer network evaluations stored in memory for the backward pass.

5 CONCLUSION

We have presented a novel method to correctly differentiate neural SDFs rendering. Unlike prior work that relies on accurate masks or biased approximations of the boundary gradients, we reparameterize the pixel filter integral to account for the discontinuities. We have validated the correctness of our approach using finite difference ground truth, and demonstrated superior optimization convergence comparing state-of-the-art neural SDF renderers.

While we have focused on primary visibility in this work, our formulation can be extended to handle global illumination. In particular, we expect to be able to model light rays and jointly optimize for SDF geometry as well as surface reflectance and illumination. Modeling full global illumination with neural SDFs may require extensions or approximations to be computationally tractable. Finally, inverse rendering under unknown, natural illumination is ill-posed and it would be interesting to explore geometry, material and illumination priors that can be combined with our differentiable rendering formulation.

REFERENCES

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

Matan Atzmon and Yaron Lipman. 2020. SAL: Sign Agnostic Learning of Shapes From Raw Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sai 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.

Sai Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David J. Kriegman, and Ravi Ramamoorthi. 2020a. Neural Reflectance Fields for Appearance Acquisition. *CoRR* abs/2008.03824 (2020).

Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020b. Deep Reflectance Volumes: Relightable Reconstructions from Multi-View Photometric Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

James F Blinn. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph.* 1, 3 (1982), 235–256.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs*. http://github.com/google/jax

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

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

Pau Gargallo, Emmanuel Prados, and Peter Sturm. 2007. Minimizing the reprojection error in surface reconstruction from images. In *International Conference on Computer Vision*. 1–8.

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse Volume Rendering with Material Dictionaries. *ACM Trans. Graph.* 32, 6 (nov 2013), 162:1–162:13.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of Machine Learning and Systems 2020*. 3569–3579.

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.

Aaron Hertzmann. 1999. Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. In *SIGGRAPH Course Notes. Course on Non-Photorealistic Rendering*, Stuart Green (Ed.). ACM Press/ACM SIGGRAPH, New York.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *symposium on User interface software and technology*. 559–568.

Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. 2020. SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization. In *Computer Vision and Pattern Recognition*. 1251–1261.

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

Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. 2021. Neural lumigraph rendering. In *Computer Vision and Pattern Recognition*. 4287–4297.

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

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

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

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. *NeurIPS* (2020).

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

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 65:1–65:14 pages.

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

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

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

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision*.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2020. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *Computer Vision and Pattern Recognition*. 3504–3515.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392406

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *International Conference on Computer Vision (ICCV)*.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Computer Vision and Pattern Recognition*. 165–174.

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A First-order Analysis of Lighting, Shading, and Shadows. *ACM Trans. Graph.* 26, 1 (2007), 2.

Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoît Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. 2020. MeshSDF: Differentiable iso-surface extraction. In *Advances in Neural Information Processing Systems*, Vol. 33. 22468–22478.

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A Versatile Scene Model with Differentiable Visibility Applied to Generative Pose Estimation. In *International Conference on Computer Vision*. IEEE, 765–773.

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

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford University. Advisor(s) Guibas, Leonidas J.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path replay backpropagation: differentiating light paths using constant memory and linear time. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (2021).

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022), 125:1–125:18. https://doi.org/10.1145/3528223.3530139

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Advances in Neural Information Processing Systems*.

Greg Ward and Paul Heckbert. 1992. Irradiance Gradients. In *Eurographics Workshop on Rendering*. Eurographics Association, 85–98.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Comput. Graph. Forum. State of the Art Report* (2022).

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems*.

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).

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo estimators for differential light transport. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (2021).

Cheng Zhang, Zhao Dong, Michael Doggett, and Shuang Zhao. 2021a. Antithetic Sampling for Monte Carlo Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (2021).

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 6 (2020), 143:1–143:19.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (2019), 227.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021c. Path-Space Differentiable Rendering of Participating Media. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (2021), 76:1–76:15.

Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021b. PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Computer Vision and Pattern Recognition*. 5453–5462.

Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. 2020. Physics-Based Differentiable Rendering: From Theory to Implementation. In *ACM SIGGRAPH 2020 Courses*. Article 14, 30 pages.

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