Universal Differentiable Renderer for Implicit Neural Representations

Lior Yariv, Matan Atzmon, and Yaron Lipman

Weizmann Institute of Science

Abstract. The goal of this work is to learn implicit 3D shape representation with 2D supervision (i.e., a collection of images). To that end we introduce the Universal Differentiable Renderer (UDR) a neural network architecture that can provably approximate reflected light from an implicit neural representation of a 3D surface, under a wide set of reflectance properties and lighting conditions. Experimenting with the task of multiview 3D reconstruction, we find our model to improve upon the baselines in the accuracy of the reconstructed 3D geometry and rendering from unseen viewing directions.

1 Introduction

Learning 3D shapes with 2D supervision (i.e., images) is a fundamental computer vision problem. A recent successful neural networks approach to solve this problem is to use a differentiable rendering system coupled with a choice of 3D geometry representation. Differential rendering systems are mostly based on ray casting/tracing [**?**,**?**,**?**,**?**,**?**,**?**], or rasterization [**?**,**?**,**?**,**?**,**?**], while popular models to represent 3D geometry include point clouds [**?**], triangle meshes [**?**], implicit representations defined over volumetric grids [**?**], and recently also neural implicit representations, namely, zero level sets of neural networks [**?**,**?**].

The main benefit in implicit neural representations is their flexibility to represent surfaces with arbitrary shape and topology, and produce smooth surface approximation without a fixed discretization, in contrast to, e.g., an implicit function defined over a volumetric grid or a triangle mesh. Thus far, differentiable rendering systems with implicit neural representations [**?**,**?**,**?**] did not incorporate lighting and reflectance properties required for producing faithful appearance of 3D geometry in images. The main challenge seems to be relating, in a differentiable way properties of the implicit surface and the parameters of the neural network representing it.

The goal of this paper is to introduce the Universal Differentiable Renderer (UDR) for implicit neural representations. That is, a differentiable renderer that is *provably* able to approximate the reflected light from a 3D shape represented as a zero level set of a neural network with an arbitrary bidirectional reflectance distribution function (BRDF) and lighting conditions from a certain family. Although the BRDF/lighting family we consider in this paper is not the most general one, as it excludes secondary lighting effects, it includes, or can approximate with arbitrary precision, many real-life appearances of 3D surfaces.

We represent the geometry as the zero level set of neural network and use the recent implicit (Eikonal) regularization [**?**] to produce an approximated signed distance function. For the UDR we require differentiable (w.r.t. the network parameters) surface points and normals and we achieve that using sample networks [**?**] and a version of automatic differentiation [**?**].

Most related to our paper is the recent work [**?**] that is first to introduce a fully differentiable renderer of implicit neural occupancy functions [**?**]. Although their model can represent arbitrary color and texture it is not universal in the sense we define, namely cannot generate arbitrary reflectance and lighting effects. In fact, we show that the model in [**?**], as-well-as several other baselines, already fail to generate the Phong reflection model [**?**]. Experimentally, we show that using the UDR in a learning framework produces more accurate 3D reconstructions of shapes from multi-view 2D supervision. Notably, while the baseline method often presents shape artifact in presence of specularity, the UDR is robust to such lighting effects.

To summarize, the key contributions of our approach are:

- Definition and implementation of Universal Differential Renderer (UDR), provably able to generate a wide range of lighting/reflectance models.
- Proving several baseline methods are not UDR.
- Using Eikonal implicit regularization to represent the geometry in the differentiable renderer as an approximate signed distance function.
- Closed-form and differentiable surface point locations and surface normals in the UDR.

2 Previous work

Related previous work of differentiable rendering systems for learning geometry comes (mostly) in two flavors: differentiable rasterization, and differentiable ray casting. Since this work falls into the second category we will concentrate on that branch of works.

2.1 Differentiable rasterization

Differentiable rasterization usually works with *explicit* shape representations such as point clouds and polygonal meshes. Since the rasterization process is not differentiable by nature most methods provide approximated or partial derivatives of pixel color with respect to shape and lighting parameters.

[**?**] made the case that approximated gradients are sufficient for differentiable renderer; [**?**] suggest approximating gradients by replacing the standard piecewise constant triangle support functions with piecewise linear functions; [**?**,**?**] suggest differentiating foreground pixel values using barycentric interpolation of vertex properties (e.g., colors, normals). [**?**,**?**] suggest a soft rasterization process: they replace the discrete rasterizartion sampling with probabilistic rendering to

achieve a differentiable process. While differentiable rasterization mostly works with traignle meshes, other representations exists, e.g., [**?**] that introduce point-based differentiable renderer by differentiating the splatting process.

2.2 Differentiable ray casting

Differentiable ray casting is mostly used with *implicit* shape representations such as implicit function defined over a volumetric grid or implicit neural representation, where the implicit function can be the occupancy function [**?**,**?**], signed distance function (SDF) [**?**] or any other signed implicit [**?**].

A related paper to ours is [**?**] that use a volumetric grid to represent an SDF and use ray casting differentiable renderer. They approximate the SDF value and its normal in each volumetric cell. In contrast, we use a neural implicit representation and therefore not restricted to a fixed grid, compute the exact surface intersection point and normal (sampled at the moving surface intersection point), and consider a more general reflection and lighting setup.

Another related paper is [**?**] that use sphere tracing of pre-trained DeepSDF model [**?**] and approximate the depth gradients w.r.t. the latent code of the DeepSDF network by differentiating the individual steps of the sphere tracing algorithm. [**?**] use LSTM for differentiable ray casting and consequently don't have an implicit surface representation per-se, i.e., the surface is not a level set of a function and can only be accessed via the ray casting algorithm. [**?**] use field probing to facilitate differentiable ray casting, and approximate derivative of the implicit function using finite differences. They also use explicit geometric regularization.

In contrast to these algorithms, our algorithm, works with exact gradients of surface points and normals of the implicit rerpesentation, uses implicit geometric regularization (Eikonal regularization), and considers a more general lighting/reflectance model.

3 Method

Given a surface represented as a the zero level set of a neural network f,

$$\mathcal{M}_\theta = \left\{ \boldsymbol{x} \in \mathbb{R}^3 \mid f(\boldsymbol{x};\theta) = 0 \right\}, \tag{1}$$

with learnable parameters $\theta \in \mathbb{R}^m$, a camera located at $\boldsymbol{c} \in \mathbb{R}^3$ and direction $\boldsymbol{v} \in \mathcal{S}$, where $\mathcal{S}$ is the unit sphere, we would like to express the pixel color $I \in [0,1]$ (assume a single channel for now), defined as the amount of light entering the camera $\boldsymbol{c}$ in direction $\boldsymbol{v}$, as a differentiable function of the surface parameters θ. Of-course, aside from the geometry of the surface $\mathcal{M}_\theta$, I depends on material properties of the surface and scene lighting.

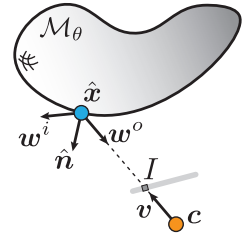


Fig. 1. Notations.

Our goal is define a *Universal Differentiable Rendered* (UDR), which is a neural network R with parameters $(\theta,\gamma) \in \mathbb{R}^{m+n}$ that satisfies the following: For arbitrary material and lighting conditions there exists $\gamma \in \mathbb{R}^n$ so that $R(\boldsymbol{c},\boldsymbol{v};\theta,\gamma)$ approximates I for different geometries $\mathcal{M}_\theta$, camera location $\boldsymbol{c}$ and direction $\boldsymbol{v}$.

3.1 Notations and the rendering equation

Let $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$ denote the first intersection of the ray $\{\boldsymbol{c} + t\boldsymbol{v} | t \geq 0\}$ with the surface $\mathcal{M}_\theta$, and $\hat{\boldsymbol{n}} = \hat{\boldsymbol{n}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$ the normal to $\mathcal{M}_\theta$ at $\hat{\boldsymbol{x}}$. The pixel color I equals the light reflected from $\mathcal{M}_\theta$ at $\hat{\boldsymbol{x}}$ in direction $-\boldsymbol{v}$ reaching $\boldsymbol{c}$. It is calculated by two functions: The bidirectional reflectance distribution function (BRDF) describing the reflectance and color properties of the surface, and the light emitted in the scene (i.e., light sources).

The BRDF function $f^r(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{w}^o, \boldsymbol{w}^i)$ describes the proportion of reflected radiance (i.e., flux of light) at some wave-length (i.e., color) leaving the surface point $\boldsymbol{x} \in \mathbb{R}^3$ with normal $\boldsymbol{n} \in \mathcal{S}$ at direction $\boldsymbol{w}^o \in \mathcal{S}$ with respect to the incoming radiance from direction $\boldsymbol{w}^i \in \mathcal{S}$. We let the BRDF depend also on the normal $\boldsymbol{n}$ to the surface at a point; BRDFs are usually defined with respect to the normal, although, usually, this dependence is not made explicit due to the standard assumption that the geometry is fixed, which isn't the case here. The light sources in the scene are described by a function $L^e(\boldsymbol{x}, \boldsymbol{w}^o)$ measuring the emitted radiance of light at some wave-length at point $\boldsymbol{x} \in \mathbb{R}^3$ in direction $\boldsymbol{w}^o \in \mathcal{S}$.

The amount of light reaching $\boldsymbol{c}$ in direction $\boldsymbol{v}$ equals the amount of light reflected from $\hat{\boldsymbol{x}}$ in direction $\boldsymbol{w}^o = -\boldsymbol{v}$ and is described by the so-called rendering equation [**?**,**?**]:

$$L(\hat{\boldsymbol{x}}, \boldsymbol{w}^o) = L^e(\hat{\boldsymbol{x}}, \boldsymbol{w}^o) + \int_\Omega f^r(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}, \boldsymbol{w}^i, \boldsymbol{w}^o) L^i(\hat{\boldsymbol{x}}, \boldsymbol{w}^i)(\hat{\boldsymbol{n}} \cdot \boldsymbol{w}^i) \, d\boldsymbol{w}^i \qquad (2)$$

where $L^i(\hat{\boldsymbol{x}}, \boldsymbol{w}^i)$ encodes the incoming radiance at $\hat{\boldsymbol{x}}$ in direction $\boldsymbol{w}^i \in \mathcal{S}$, and the term $\hat{\boldsymbol{n}} \cdot \boldsymbol{w}^i$ compensates for the fact that the light does not hit the surface orthogonally; Ω is the half sphere defined by $\hat{\boldsymbol{n}}$; the rendering equation holds for every light wave-length; as described later we will use it for the red, green and blue (RGB) wave-lengths.

3.2 Restricted BRDF and lighting model

We restrict our material and lighting settings to those represented by a volumetric (continuous) BRDF function $f^r : \mathbb{R}^3 \times \mathcal{S} \times \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ and light radiance functions $L^e, L^i : \mathbb{R}^3 \times \mathcal{S} \to \mathbb{R}$. We denote the collection of such continuous functions by $\mathcal{P} = \left\{(f^r, L^e, L^i)\right\}$. This model includes many common materials and lighting conditions such as the popular Phong model [**?**]:

$$L(\hat{\boldsymbol{x}}, \boldsymbol{w}^o) = k_d O_d I_a + k_d O_d I_d \left(\hat{\boldsymbol{n}} \cdot \frac{\boldsymbol{\ell} - \hat{\boldsymbol{x}}}{\|\boldsymbol{\ell} - \hat{\boldsymbol{x}}\|} \right)_+ + k_s O_s I_d \left(\hat{\boldsymbol{r}} \cdot \frac{\boldsymbol{\ell} - \hat{\boldsymbol{x}}}{\|\boldsymbol{\ell} - \hat{\boldsymbol{x}}\|} \right)_+^{n_s}, \quad (3)$$

where $(a)_+ = \max\{a, 0\}$; k_d, k_s are the diffuse and specular coefficients, I_a, I_d are the ambient and point light source colors, O_d, O_s are the diffuse and specular colors of the surface, $\boldsymbol{\ell} \in \mathbb{R}^3$ is the location of a point light source, $\hat{\boldsymbol{r}} = -(\boldsymbol{I} - 2\hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T)\boldsymbol{w}^o$ the reflection of the viewing direction $\boldsymbol{w}^o = -\boldsymbol{v}$ with respect to the normal $\hat{\boldsymbol{n}}$, and n_s is the specular exponent.

The family $\mathcal{P}$, however, does not include all possible lighting conditions, e.g., it excludes self-shadows and second order (or higher) light reflections as L^i is independent of the geometry $\mathcal{M}_\theta$.

3.3 Universal Differentiable Renderer

Our goal is to learn geometry $\mathcal{M}_\theta$ in a scene via rendered images of this geometry under a wide range of materials and lighting conditions. We would like to be able to express these possible appearances of $\mathcal{M}_\theta$ with arbitrary material/lighting models in $\mathcal{P}$. For that end we define:

Definition 1. *Given a parametric surface family $\mathcal{M}_\theta$, $\theta \in \mathbb{R}^m$, a* Universal Differentiable Renderer *(UDR) for $\mathcal{M}_\theta$ is a function*

$$R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma)$$

differentiable in θ, γ, where $\theta \in \mathbb{R}^m$, $\gamma \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^3$, and $\boldsymbol{v} \in \mathcal{S}$, such that for arbitrary material/lighting $(f^r, L^i, L^e) \in \mathcal{P}$ there exists $\gamma \in \mathbb{R}^n$ so that $R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma)$ is an arbitrary good approximation of the light amount enters camera $\boldsymbol{c} \in \mathbb{R}^3$ in direction $\boldsymbol{v} \in \mathcal{S}$, i.e., $L(\hat{\boldsymbol{x}}, -\boldsymbol{v})$, for all $\theta, \boldsymbol{c}, \boldsymbol{v}$ in some compact domain.

Given a UDR R, we can write down a loss consisting of terms of the form

$$\text{loss}_{\text{R}}(\boldsymbol{c}, \boldsymbol{v}, \theta, \gamma) = \left| R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma) - I \right|. \tag{4}$$

The fact that R is a UDR will guarantee that we could match the appearance of geometry $\mathcal{M}_\theta$ to a wide range of materials and lighting conditions. Furthermore, the requirement that R can approximate the rendered light function for arbitrary camera, direction and geometry will facilitate separation of geometry and material/lighting in the learning process.

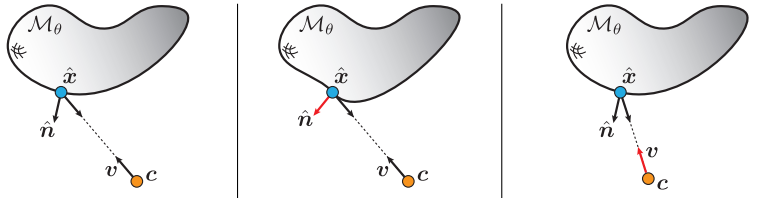
We will prove the following is a UDR for geometry $\mathcal{M}_\theta$:

$$R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma) = M(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}, \boldsymbol{v}; \gamma), \tag{5}$$

where M is an MLP with parameters $\gamma \in \mathbb{R}^n$, and $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$, $\hat{\boldsymbol{n}} = \hat{\boldsymbol{n}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$, as defined above.

Theorem 1. *The renderer in equation* **??** *is a UDR.*

Proof. Let $\mathcal{M}_\theta$ be a parametric family of surfaces. Pick an arbitrary material/lighting $(f^r, L^e, L^i) \in \mathcal{P}$. We need to show there is an MLP M so that equation **??** is an arbitrary good approximation of $L(\hat{\boldsymbol{x}}, -\boldsymbol{v})$. Equation **??** implies that there exists a continuous function F so that $L(\hat{\boldsymbol{x}}, -\boldsymbol{v}) = F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}, \boldsymbol{v})$ for all $\theta, \boldsymbol{c}, \boldsymbol{v}$. Assuming $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}, \boldsymbol{v})$ are contained in some compact set $\mathcal{K}$, we can approximate (arbitrarily well) F over $\mathcal{K}$ with an MLP M using the universality theorems of MLPs, see [**?**] and [**?**] for approximation including derivatives. □



(a) reference scene (b) change in $\hat{\boldsymbol{n}}$ (c) change in $\boldsymbol{c}$ and $\boldsymbol{v}$

Fig. 2. Differentiable renderers R without $\hat{\boldsymbol{n}}$ and/or $\boldsymbol{v}$ are not universal; changing the surface normal (b) and/or the viewing direction (c) will produce different amount of light arriving the camera $\boldsymbol{c}$ at direction $\boldsymbol{v}$ and cannot be captured by R.

An important question is: Are $\hat{\boldsymbol{n}}, \boldsymbol{v}$ necessary in equation **??** in order to achieve a UDR? Previous works, e.g., [**?**], have considered rendering functions of the form

$$R(\theta, \boldsymbol{c}, \boldsymbol{v}; \gamma) = M(\hat{\boldsymbol{x}}; \gamma). \tag{6}$$

We will next prove this model is not a universal renderer. We will in fact show that removing $\hat{\boldsymbol{n}}$ and/or $\boldsymbol{v}$ from equation **??** will result in a non-universal renderer. Assume $\mathcal{M}_\theta$ is expressive enough to change the normal direction $\hat{\boldsymbol{n}}$ arbitrarily at a point $\hat{\boldsymbol{x}}$; for example, even a linear classifier $f(\boldsymbol{x}; \boldsymbol{a}, b) = \boldsymbol{a}^T\boldsymbol{x} + b$ would do. Let $M(\hat{\boldsymbol{x}}; \gamma)$ be a UDR for $\mathcal{M}_\theta$. Choose the Phong model (equation **??**) and let $R(\theta, \boldsymbol{c}, \boldsymbol{v}; \gamma) = M(\hat{\boldsymbol{x}}, \boldsymbol{v}; \gamma)$ (i.e., remove the normal component from the renderer) be the light amount arriving at $\boldsymbol{c}$ in direction $\boldsymbol{v}$ as approximated by the renderer. Consider the setting shown in Figure **??** (a) and (b): In both cases $M(\hat{\boldsymbol{x}}, \boldsymbol{v}; \gamma)$ yields the same value although changing the normal direction at $\hat{\boldsymbol{x}}$ will produce, under the Phong model, a different light amount at $\boldsymbol{c}$.

Similarly, consider a renderer with the viewing direction ($\boldsymbol{v}$) removed, i.e., $M(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}; \gamma)$, and Figure **??** (a) and (c): In both cases $M(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}; \gamma)$ produces the same value although, under the Phong model, the reflected light can change when the point of view changes. That is, we can choose light position $\boldsymbol{\ell}$ in equation **??** so that different amount of light reaches $\boldsymbol{c}$ at direction $\boldsymbol{v}$.

3.4 Implementation of the UDR

To implement the UDR in equation **??** we first represent the geometry $\mathcal{M}_\theta$ as in equation **??** using an MLP $f : \mathbb{R}^m \to \mathbb{R}$. We enforce f to be approximately signed distance function with Implicit Geometric Regularization (IGR) [**?**], i.e., incorporating the Eikonal regularization term

$$\text{loss}_{\text{E}}(\theta) = \mathbb{E}_{\boldsymbol{x}}\big(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; \theta)\| - 1\big)^2 \tag{7}$$

where $\boldsymbol{x}$ is distributed uniformly in a bounding box of the scene.

Next, we need to have $\hat{\boldsymbol{x}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$, and $\hat{\boldsymbol{n}}(\boldsymbol{c}, \boldsymbol{v}, \theta)$ both differentiable in θ. $\hat{\boldsymbol{x}}$ requires representing the location of a point on $\mathcal{M}_\theta$ using the parameters θ of

f. This was recently done in [**?**,**?**]; we employ the sample network idea from [**?**]. That is, as we show in the supplementary material, let θ_0 denote the current parameter, a first-order approximation to $\hat{\boldsymbol{x}}$ is

$$\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta) = \hat{\boldsymbol{x}}_0 - \boldsymbol{u} f(\hat{\boldsymbol{x}}_0;\theta) \tag{8}$$

where $\hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta_0)$, and $\boldsymbol{u} = \frac{\boldsymbol{v}}{\nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}_0;\theta_0)\cdot\boldsymbol{v}}$ are constant vectors. The benefit in equation **??** is that it can be implemented by adding a single linear layer to the MLP f; $\hat{\boldsymbol{x}}_0$ is computed using the sphere tracing algorithm similar to [**?**]. The term $\hat{\boldsymbol{n}}(\boldsymbol{c},\boldsymbol{v},\theta)$ is the normal to $\mathcal{M}_\theta$ at point $\hat{\boldsymbol{x}}$. Since f is approximately a sign distance function we set

$$\hat{\boldsymbol{n}}(\boldsymbol{c},\boldsymbol{v},\theta) = \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta);\theta). \tag{9}$$

The term $\nabla_{\boldsymbol{x}} f(\boldsymbol{x};\theta)$ in equations **??** and **??** is implemented using a version of automatic differentiation of the MLP f, by building a networks that computes $(f(\boldsymbol{x};\theta), \nabla_{\boldsymbol{x}} f(\boldsymbol{x};\theta))$ in the forward-pass, as suggested in [**?**]. Lastly, we incorporate equations **??**-**??** with an MLP M to get our UDR, equation **??**; M outputs 3 RGB values.

3.5 Loss function for multi-view reconstruction

Let $I_j \in [0,1]^3$, $\alpha_j \in [0,1]$ be the RGB values and the opacity value corresponding to a pixel in an image taken with camera $\boldsymbol{c}_j \in \mathbb{R}^3$ and direction $\boldsymbol{v}_j \in \mathcal{S}$ where $j \in J$, see Figure **??**. That is, J indexes all the pixels in a collection of images.

Let $f(\boldsymbol{x};\theta)$ be an MLP where $\mathcal{M}_\theta$ represents the reconstructed surface with current parameter $\theta = \theta_0$. For each $j \in J$ we perform sphere-tracing [**?**] to compute $\hat{\boldsymbol{x}}_j = \hat{\boldsymbol{x}}_j(\boldsymbol{c}_j,\boldsymbol{v}_j,\theta_0)$. Let $J^{\text{in}} \subset J$ be the subset of indices where $\hat{\boldsymbol{x}}_j$ was found; $J \setminus J^{\text{in}}$ are therefore the indices of pixels where the ray $\{\boldsymbol{c}_j + t\boldsymbol{v}_j | t \geq 0\}$ did not intersect $\mathcal{M}_\theta$. Then our loss is

$$\begin{aligned}\text{loss}(\theta,\gamma) = &\sum_{j\in J^{\text{in}}} \Big(\alpha_j \text{loss}_{\text{R}}(\boldsymbol{c}_j,\boldsymbol{v}_j,\theta,\gamma) - \tau(1-\alpha_j) f(\boldsymbol{y}_j;\theta)\Big) \\ &+ \tau \sum_{j\in J\setminus J^{\text{in}}} \alpha_j f(\boldsymbol{z}_j;\theta) + \lambda\, \text{loss}_{\text{E}}(\theta),\end{aligned} \tag{10}$$

where $\lambda, \tau > 0$ are constant parameters, loss_{E} is defined in equation **??**, loss_{R} is defined in equation **??**, $\boldsymbol{y}_j$ is the point along the ray $\{\hat{\boldsymbol{x}}_j + t\boldsymbol{v} | t \geq 0\}$ that minimizes $f(\cdot\,;\theta)$, and $\boldsymbol{z}_j$ is the point along the ray $\{\boldsymbol{c}_j + t\boldsymbol{v} | t \geq 0\}$ that minimizes $f(\cdot\,;\theta)$; both $\boldsymbol{y}_j, \boldsymbol{z}_j$ are independent of θ. The two terms including $\boldsymbol{y}_j, \boldsymbol{z}_j$ are treating cases where the opacity coefficient $\alpha_j, (1-\alpha_j)$ is in conflict with the existent/nonexistent intersection of the ray $\{\boldsymbol{c} + t\boldsymbol{v} | t \geq 0\}$ and $\mathcal{M}_\theta$.

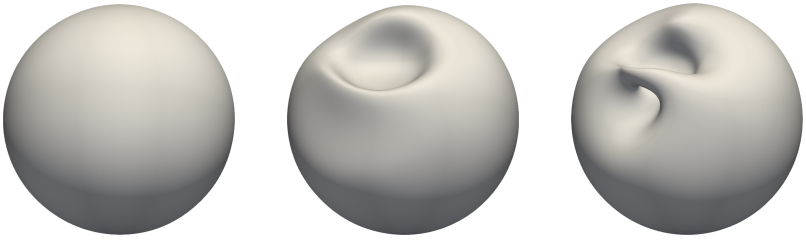


Fig. 3. Shiny sphere test (Phong reflection model). The geometry $\mathcal{M}_\theta$ reconstructed by the UDR, DVR, and DRnoV models.

4 Experiments

We compared our UDR model, equation **??**, to the Differentiable Volumetric Rendering (DVR) [**?**], namely $R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma) = M(\hat{\boldsymbol{x}}; \gamma)$, and to a model similar to UDR but without the view direction (DRnoV), $R(\boldsymbol{c}, \boldsymbol{v}; \theta, \gamma) = M(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}; \gamma)$.

Implementation details. We represent the geometry $\mathcal{M}_\theta$ using an MLP $f(\boldsymbol{x}; \theta)$, $f : \mathbb{R}^3 \times \mathbb{R}^m \to \mathbb{R}$, consisting of 8 layers with hidden dimension of size 512, and a single skip connection from the input to the middle layer as in [**?**]. We initialize the weights $\theta \in \mathbb{R}^m$ using the geometric initialization from [**?**]; this provides a close to sign distance function already at initialization. For the implementation of the UDR we use also an MLP, $M(\hat{\boldsymbol{x}}, \hat{\boldsymbol{n}}, \boldsymbol{v}; \gamma)$, $M : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n \to \mathbb{R}^3$, consisting of of 4 layers, with 512 hidden units. In our loss, equation **??**, we set the constant parameters to be $\lambda = 0.1$ and $\tau = 5$.

4.1 Shiny sphere test

We evaluate the different differentiable renderers with a simple sphere geometry with a strong specularity generated with a Phong reflection model (equation **??**). We have generated 50 train images of size 256×256 pixels from random camera locations pointing toward the sphere's center and trained geometry $\mathcal{M}_\theta$ and the different renderers R via minimization of the loss in equation **??**. Figure **??** depicts the reconstructed geometries $\mathcal{M}_\theta$ with the different renderers: UDR, DrnoV, and DVR; Figure **??** depicts images from unseen camera locations and directions created using the trained geometry and different renderers. As can be observed, and as suggested by the theory in Section **??**, UDR is able to separate well geometry and reflectance, reproducing the sphere shape and near perfect renderings; the baselines, on the other hand, tend to confuse lighting and geometry and do not reconstruct the sphere shape or the lighting properties of the scene.

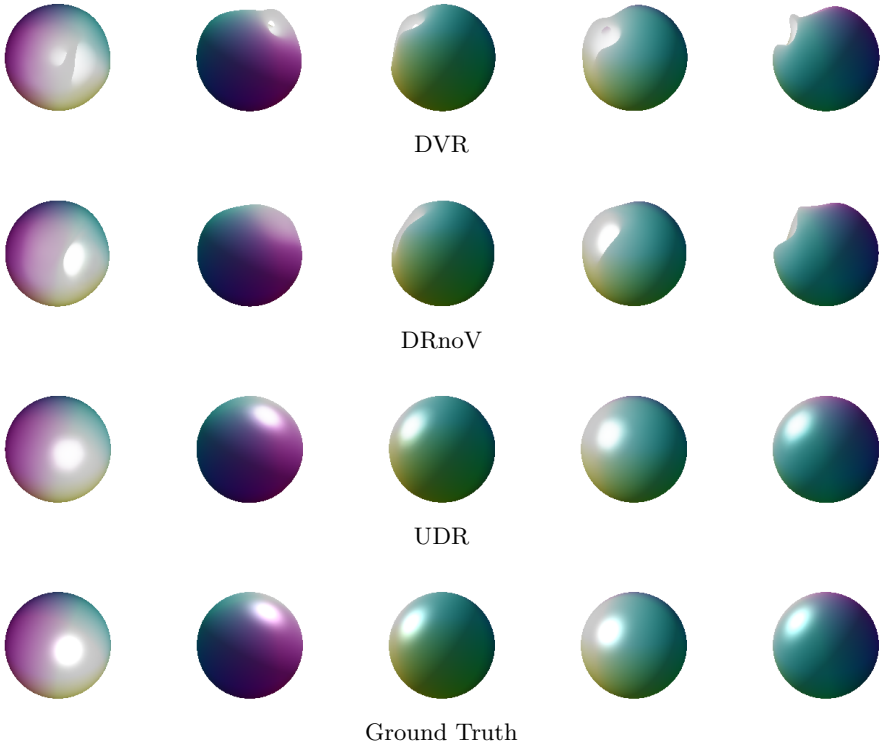


Fig. 4. Shiny sphere test (Phong reflection model). Columns represent renderings from unseen camera locations and directions. Each row is produced by (from top to bottom): DVR, DRnoV, UDR and ground truth. Note that as predicted from the theory, only UDR is able to render faithful images of the shiny sphere.

4.2 3D reconstruction with multiview supervision

In this experiment we tested reconstruction of 3D models using the image dataset of Choy et al.[**?**]. This dataset contains 13 model categories from the ShapeNet dataset [**?**], where each model has 24 rendered images of size 137×137 with known camera parameters and opacity channel. We have taken the first 10 models in each category and for each model we tested each of the renderers mentioned above. That is, for each model, we extracted the per-pixel information $(I_j, \alpha_j, \boldsymbol{c}_j, \boldsymbol{v}_j)$, $j \in J$, from the 24 images, and optimized the loss in equation **??** to find the geometry $\mathcal{M}_\theta$ and renderer network M for each model.

Evaluation. We evaluate the accuracy of the 3D surface reconstruction ($\mathcal{M}_\theta$) and rendered images of the different renderers (R) from the input directions. The 3D reconstruction accuracy is measured with the standard Chamfer-L_2 distance:

$$d_{\mathrm{C}}\left(\mathcal{X}_1, \mathcal{X}_2\right) = d\left(\mathcal{X}_1, \mathcal{X}_2\right) + d\left(\mathcal{X}_2, \mathcal{X}_1\right),$$

Category	Chamfer-L_2 DVR [?]	Chamfer-L_2 DRnoV	Chamfer-L_2 UDR	PSNR DVR [?]	PSNR DRnoV	PSNR UDR
Airplane	0.72 ± 0.71	3.28 ± 3.47	$\mathbf{0.55} \pm 0.16$	24.89 ± 2.13	23.88 ± 3.54	$\mathbf{25.47} \pm 2.57$
Bench	3.45 ± 2.09	3.64 ± 3.24	$\mathbf{1.8} \pm 1.61$	19.89 ± 4.39	21.26 ± 4.93	$\mathbf{22.61} \pm 3.58$
Cabinet	1.63 ± 1.19	1.63 ± 1.07	$\mathbf{1.25} \pm 1.05$	23.9 ± 3.88	24.11 ± 4.19	$\mathbf{25.83} \pm 3.89$
Car	$\mathbf{1.15} \pm 0.34$	1.36 ± 0.77	1.53 ± 0.83	19.8 ± 1.58	$\mathbf{21.15} \pm 1.91$	21.14 ± 1.67
Chair	4.63 ± 5.62	3.06 ± 3.74	$\mathbf{1.95} \pm 3.09$	20.21 ± 3.98	20.94 ± 4.14	$\mathbf{22.01} \pm 3.97$
Display	0.95 ± 0.68	1.21 ± 1.58	$\mathbf{0.67} \pm 0.26$	22.59 ± 4.64	23.76 ± 3.91	$\mathbf{25.22} \pm 3.86$
Lamp	3.04 ± 3.0	2.3 ± 1.66	$\mathbf{1.31} \pm 1.04$	22.29 ± 3.54	22.13 ± 3.46	$\mathbf{23.62} \pm 3.17$
Speaker	$\mathbf{1.79} \pm 1.04$	1.95 ± 1.69	2.49 ± 2.62	$\mathbf{23.03} \pm 3.15$	23.31 ± 3.63	23.69 ± 4.79
Rifle	0.42 ± 0.3	$\mathbf{0.35} \pm 0.29$	0.73 ± 0.64	23.29 ± 2.58	$\mathbf{23.63} \pm 2.81$	23.24 ± 2.81
Sofa	2.02 ± 1.6	3.07 ± 2.76	$\mathbf{1.42} \pm 0.8$	23.03 ± 2.68	23.0 ± 3.05	$\mathbf{25.21} \pm 3.14$
Table	3.3 ± 2.19	3.27 ± 1.93	$\mathbf{1.06} \pm 1.06$	20.43 ± 3.7	21.59 ± 4.91	$\mathbf{23.97} \pm 3.59$
Telephone	0.7 ± 1.02	$\mathbf{0.34} \pm 0.28$	0.4 ± 0.43	23.34 ± 4.34	23.76 ± 3.61	$\mathbf{24.37} \pm 3.0$
Vessel	1.72 ± 2.05	1.33 ± 1.59	$\mathbf{0.96} \pm 0.8$	24.82 ± 2.6	25.62 ± 2.79	$\mathbf{26.45} \pm 2.63$

Table 1. Multiview 3D reconstruction, quantitative results. We report L_2 Chamfer distance (multiplied by 10^3, mean $\pm$ std) between the reconstructed 3D surface and the ground truth; and PSNR (in dB, mean $\pm$ std) of the rendered images using the trained renderers and input images.

where

$$d\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{|\mathcal{X}_1|} \sum_{\boldsymbol{x}_1 \in \mathcal{X}_1} \min_{\boldsymbol{x}_2 \in \mathcal{X}_2} \left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|,$$

and $\|\cdot\|$ is the 2-norm. We also measure the PSNR of our renderings compared to the input 24 images. Table **??** log the results; Figures **??**, **??** depict the ground truth models, reconstructed 3D geometry with DVR and UDR, examples of the input images, and rendered (higher-res) images from unseen directions using the trained renderers. Note that our method compares favorably to the baseline and provide more accurate 3D reconstructions as well as renderings from unseen directions.

4.3 Method limitations

Figure **??** shows some typical failure cases, illustrating the difficulties of our method with 3D reconstructions of thin objects, and generalization to unseen parts. The generalization to unseen parts could be potentially resolved by learning shape space, collecting information of the shape from different by similar shapes. Lastly, note the low resolution of the input images, a fact which can explain some of artifacts in out results.

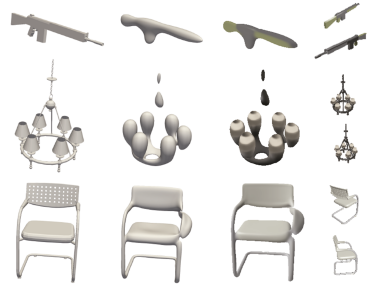


Fig. 5. Failure cases of UDR. From left to right: ground truth, UDR reconstruction, UDR rendering, and example input images.

5 Conclusions

We have introduced the Universal Differentiable Renderer (UDR) that, as far as we are aware, is the first neural network architecture that can provably approximate a wide range of appearances (material/lighting) of 3D geometries represented as zero level sets of neural networks. A current limitation of the method seems to be the rather heavy computational complexity of the training, requires computing sphere tracings of many pixels in each iteration of the algorithm. Incorporating different acceleration techniques and/or importance sampling could alleviate the situation and we mark it as future work. Another future work direction is to incorporate the UDR in other computer vision and learning applications such as 3D model generation, structure from motion, and learning 3D models from images without known camera properties.

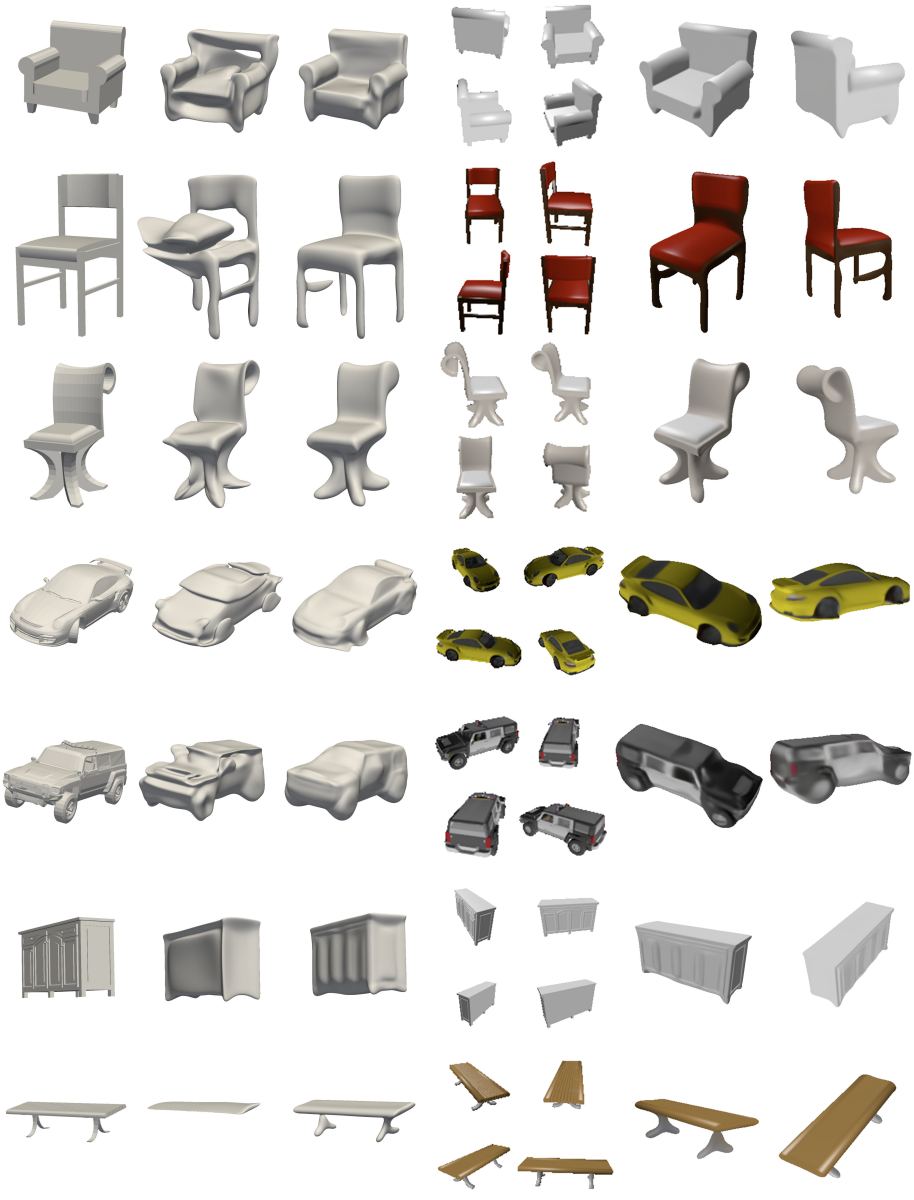


Fig. 6. Multiview 3D reconstruction, qualitative results (A). We show (left to right): ground truth geometry, DVR reconstruction, UDR reconstruction, examples of input images, and 2 renderings from unseen directions.

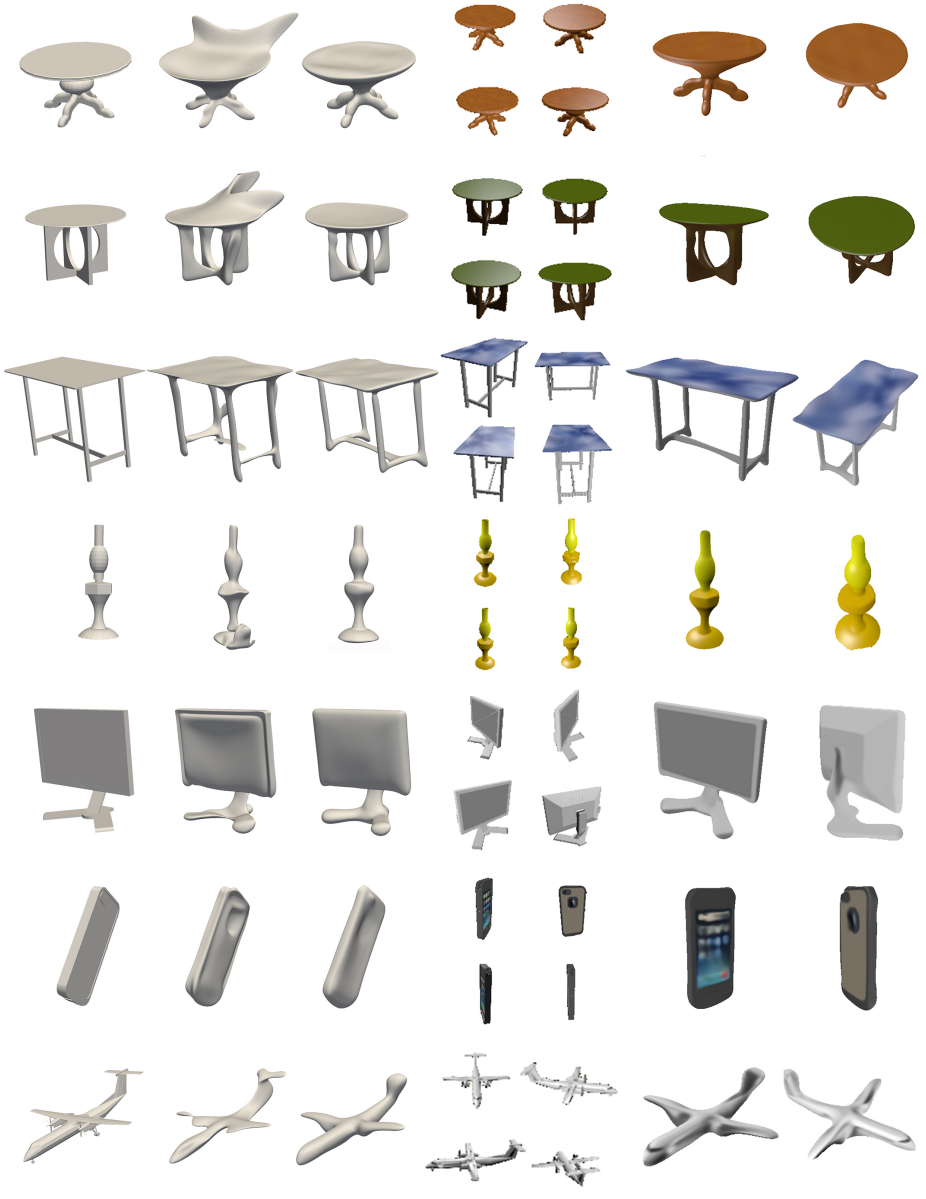


Fig. 7. Multiview 3D reconstruction, qualitative results (B). We show (left to right): ground truth geometry, DVR reconstruction, UDR reconstruction, examples of input images, and 2 renderings from unseen directions.

References

1. Atzmon, M., Haim, N., Yariv, L., Israelov, O., Maron, H., Lipman, Y.: Controlling neural level sets. In: Advances in Neural Information Processing Systems. pp. 2032–2041 (2019)
2. Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. arXiv preprint arXiv:1911.10414 (2019)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
4. Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to predict 3d objects with an interpolation-based differentiable renderer. In: Advances in Neural Information Processing Systems. pp. 9605–9616 (2019)
5. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019)
6. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
7. Foley, J.D., Van, F.D., Van Dam, A., Feiner, S.K., Hughes, J.F., Angel, E., Hughes, J.: Computer graphics: principles and practice, vol. 12110. Addison-Wesley Professional (1996)
8. Genova, K., Cole, F., Maschinot, A., Sarna, A., Vlasic, D., Freeman, W.T.: Unsupervised training for 3d morphable model regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8377–8386 (2018)
9. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes (2020)
10. Hart, J.C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer **12**(10), 527–545 (1996)
11. Hornik, K., Stinchcombe, M., White, H.: Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural networks **3**(5), 551–560 (1990)
12. Hornik, K., Stinchcombe, M., White, H., et al.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
13. Immel, D.S., Cohen, M.F., Greenberg, D.P.: A radiosity method for non-diffuse environments. Acm Siggraph Computer Graphics **20**(4), 133–142 (1986)
14. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. arXiv preprint arXiv:1912.07109 (2019)
15. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 143–150 (1986)
16. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3907–3916 (2018)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. ACM Transactions on Graphics (TOG) **37**(6), 1–11 (2018)

19. Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., Cui, Z.: Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. arXiv preprint arXiv:1911.13225 (2019)
20. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. arXiv preprint arXiv:1901.05567 (2019)
21. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7708–7717 (2019)
22. Liu, S., Saito, S., Chen, W., Li, H.: Learning to infer implicit surfaces without 3d supervision. In: Advances in Neural Information Processing Systems. pp. 8293–8304 (2019)
23. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer. In: European Conference on Computer Vision. pp. 154–169. Springer (2014)
24. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
25. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. arXiv preprint arXiv:1912.07372 (2019)
26. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
27. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
28. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2304–2314 (2019)
29. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems. pp. 1119–1130 (2019)
30. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (TOG) **38**(6), 1–14 (2019)

6 Supplementary Material

6.1 Directional Sample Network

Let the point $\boldsymbol{x}$ be the intersection of $\mathcal{M}_\theta$ and the ray $\{\boldsymbol{c}+t\boldsymbol{v}|t\geq 0\}$, where $\boldsymbol{c}\in\mathbb{R}^3$ is the camera center and $\boldsymbol{v}\in\mathcal{S}$ is the viewing direction. That is

$$\boldsymbol{x}(\boldsymbol{c},\boldsymbol{v},\theta)=\mathcal{M}_\theta\cap\{\boldsymbol{c}+t\boldsymbol{v}\mid t\geq 0\}\,. \tag{11}$$

We want to produce a first order approximation $\hat{\boldsymbol{x}}$ to $\boldsymbol{x}$ at $\theta=\theta_0\in\mathbb{R}^m$, that is a neural network $\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta)$ so that

$$\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta_0)=\hat{\boldsymbol{x}}_0, \tag{12}$$

where $\hat{\boldsymbol{x}}_0=\boldsymbol{x}(\boldsymbol{c},\boldsymbol{v},\theta_0)$, and

$$\nabla_\theta\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta_0)=\nabla_\theta\boldsymbol{x}(\boldsymbol{c},\boldsymbol{v},\theta_0). \tag{13}$$

We will use the sample network from [**?**]. In particular, we prove that equation **??** in the main paper, namely the network constructed by adding a single linear layer to the MLP f,

$$\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta)=\hat{\boldsymbol{x}}_0-\boldsymbol{u}f(\hat{\boldsymbol{x}}_0;\theta) \tag{14}$$

$\boldsymbol{u}=\frac{\boldsymbol{v}}{\nabla_{\boldsymbol{x}}f(\hat{\boldsymbol{x}}_0;\theta_0)\cdot\boldsymbol{v}}$, satisfies these conditions.

First, equation **??** holds since $f(\hat{\boldsymbol{x}}_0;\theta_0)=0$. Second, to show equation **??** we note that $f(\boldsymbol{x};\theta)\equiv 0$. Using the chain rule and setting $\theta=\theta_0$ we get:

$$\frac{\partial f(\hat{\boldsymbol{x}}_0;\theta_0)}{\partial\theta}+\nabla_{\boldsymbol{x}}f(\hat{\boldsymbol{x}}_0;\theta_0)^T\frac{\partial\boldsymbol{x}(\boldsymbol{c},\boldsymbol{v},\theta_0)}{\partial\theta}=0. \tag{15}$$

Next, by equation **??**, $\frac{\partial\boldsymbol{x}(\boldsymbol{c},\boldsymbol{v},\theta_0)}{\partial\theta}=\boldsymbol{v}\boldsymbol{w}^T$, for some $\boldsymbol{w}\in\mathbb{R}^m$. Plugging this back in equation **??** and solving for $\boldsymbol{w}$ yields

$$\boldsymbol{w}^T=-\frac{1}{\nabla_{\boldsymbol{x}}f(\hat{\boldsymbol{x}}_0;\theta_0)\cdot\boldsymbol{v}}\frac{\partial f(\hat{\boldsymbol{x}}_0;\theta_0)}{\partial\theta}.$$

Finally, notice that $\frac{\partial\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta_0)}{\partial\theta}=\boldsymbol{v}\boldsymbol{w}^T$.

Relation to [?]. Implicit differentiation, as in equation **??**, has been exploited recently also in [**?**]. However, our implementation differs in deriving the sample network in equation **??**; the benefit in the sample network is that it is easily implemented by adding a fixed linear layer to the MLP f for every sample point $\hat{\boldsymbol{x}}_0$, see [**?**] for more details. As the UDR model (see equation **??**) additionally requires surface normals, our use of the sample network allows us to achieve a differentiable normal representation, $\hat{\boldsymbol{n}}(\boldsymbol{c},\boldsymbol{v},\theta)=\nabla_{\boldsymbol{x}}f(\hat{\boldsymbol{x}}(\boldsymbol{c},\boldsymbol{v},\theta);\theta)$, as well. Note that $\nabla_{\boldsymbol{x}}f$ represents the normal direction of the level sets of a signed distance function (SDF); one could explicitly normalize $\nabla_{\boldsymbol{x}}f/\left\|\nabla_{\boldsymbol{x}}f\right\|$ to get the precise normal, even for approximate or non-SDF; however, we worked with $\hat{\boldsymbol{n}}$ as detailed.

6.2 Implementation Details

Sphere tracing. Given a camera position $\boldsymbol{c}$ and a direction $\boldsymbol{v} \in \mathcal{S}$ that together characterize some pixel location, we performed sphere tracing along the ray $\{\boldsymbol{c} + t\boldsymbol{v} \mid t \geq 0\}$ to find the corresponding surface sample $\hat{\boldsymbol{x}}_0$ from $\mathcal{M}_\theta$. As $\mathcal{M}_\theta$ is defined by an approximate sign distance field f we opted for sphere tracing algorithm [**?**]. In Algorithm **??** we describe the details of our implementation for the sphere tracing algorithm.

Algorithm 1: Sphere tracing

Data: Initial point $\boldsymbol{p}_0$, ray direction $\boldsymbol{v}$
Result: The intersection $\boldsymbol{x} = \mathcal{M}_{\theta_0} \cap \{\boldsymbol{p}_0 + t\boldsymbol{v} | t \geq 0\}$

$\boldsymbol{x} = \boldsymbol{p}_0$;
while $f(\boldsymbol{x}; \theta_0) > \epsilon$ *and* $\|\boldsymbol{x}\|_2 < r$ **do**
  $d = f(\boldsymbol{x}; \theta_0)$;
  if $f(\boldsymbol{x} + d\boldsymbol{v}; \theta_0) < 0$ **then**
    $d = \alpha \cdot d$
  end
  $\boldsymbol{x} = \boldsymbol{x} + d\boldsymbol{v}$;
end

The algorithm uses the following parameters: $\epsilon > 0$ is the convergence threshold, and $r > 0$ is the radius of the bounding sphere of the object (we assume the objected is centered at the origin). For all of our experiments we set $\epsilon = 1\text{e}{-5}$, and $r = 1$. Moreover, we set $\alpha = 0.5$, used to make conservative distance estimations in case f is not a perfect signed distance function. $\boldsymbol{p}_0$ is the initialized point for the algorithm and usually defined as the camera center, $\boldsymbol{p}_0 = \boldsymbol{c}$. In order to accelerate the process we took $\boldsymbol{p}_0$ as the first intersection point of the ray $\{\boldsymbol{c} + t\boldsymbol{v} | t \geq 0\}$ and the bounding sphere $\{\boldsymbol{x} \,|\, \|\boldsymbol{x}\|_2 \leq r\}$, as done in [**?**]. We parallelized this process on the GPU for multiple views and directions $\{(\boldsymbol{c}_j, \boldsymbol{v}_j)\}$.

Training Details. We trained using the ADAM optimizer [**?**] with 1e−4 as our learning rate, setting for each step the number of views to 24 and the number of sampled pixels to 1024. All models were trained for 10K epochs. Training was done on a single Nvidia V-100 GPU, using PYTORCH deep learning framework [**?**].
Regarding the loss in equation **??**, we find $\boldsymbol{y}_j$ and $\boldsymbol{z}_j$ by randomly sampling the ray $\{\hat{\boldsymbol{x}}_j + t\boldsymbol{v} | t \geq 0\}$ and $\{\boldsymbol{c}_j + t\boldsymbol{v} | t \geq 0\}$ accordingly, and took the points that minimize $f(\cdot\,; \theta)$.

6.3 Additional Results

Figure **??** depicts additional results from the multiview 3D reconstruction experiment described in section **??**.

Fig. 8. Multiview 3D reconstruction, qualitative results (C). We show (left to right): ground truth geometry, DVR reconstruction, UDR reconstruction, examples of input images, and 2 renderings from unseen directions.