



# A Level Set Theory for Neural Implicit Evolution Under Explicit Flows

Ishit Mehta<sup>(✉)</sup>, Manmohan Chandraker, and Ravi Ramamoorthi

University of California San Diego, La Jolla, USA  
mehtaib@gmail.com

**Abstract.** Coordinate-based neural networks parameterizing implicit surfaces have emerged as efficient representations of geometry. They effectively act as parametric level sets with the zero-level set defining the surface of interest. We present a framework that allows applying deformation operations defined for triangle meshes onto such implicit surfaces. Several of these operations can be viewed as energy-minimization problems that induce an instantaneous flow field on the explicit surface. Our method uses the flow field to deform parametric implicit surfaces by extending the classical theory of level sets. We also derive a consolidated view for existing methods on differentiable surface extraction and rendering, by formalizing connections to the level-set theory. We show that these methods drift from the theory and that our approach exhibits improvements for applications like surface smoothing, mean-curvature flow, inverse rendering and user-defined editing on implicit geometry.

**Keywords:** Implicit surfaces · Level sets · Euler-lagrangian deformation

## 1 Introduction

Recent successes in generative modeling of shapes [13, 46, 52] and inverse rendering [41, 69] are largely driven by implicit representations of geometry parameterized as multi-layer perceptrons (MLPs) (or neural implicits [16]). These networks can compactly represent highly-detailed surfaces at (theoretically) infinite resolution [34, 53, 59, 60]; they are defined continuously in  $\mathbb{R}^3$  and are differentiable – enabling their usage in gradient-based optimization [25, 69] and learning [2, 12, 37] methods. Despite these advances, there is still a large body of work in geometry processing, computer vision and graphics which relies on explicit surface representations. Often these mesh-based algorithms are a better choice than their implicit counterparts. For instance, in case of inverse rendering, differentiable renderers for triangle meshes [18, 26, 27, 42, 71] are a) faster, b) more accurate, and c) can handle more complex light-transport effects, in comparison to the

---

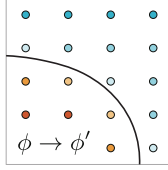
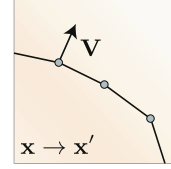
**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-031-20086-1\\_41](https://doi.org/10.1007/978-3-031-20086-1_41).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022  
S. Avidan et al. (Eds.): ECCV 2022, LNCS 13662, pp. 711–729, 2022.  
[https://doi.org/10.1007/978-3-031-20086-1\\_41](https://doi.org/10.1007/978-3-031-20086-1_41)

renderers designed for implicit surfaces [23, 41, 69]. Similarly, geometry processing algorithms for applications like surface smoothing and deformation [10, 54, 61] are vastly superior in terms of compute and memory requirements than the ones developed for neural implicit surfaces [67]. Most of these methods, however, are highly specific to mesh geometry and are not easily adaptable to MLP-defined surfaces. Our work is a theoretical attempt to bridge this gap.

We first introduce the following insight: several mesh-based algorithms define an energy-minimization problem that is solved using gradient descent; the gradients used by the optimizer to update the geometry can be viewed as analogous to an instantaneous explicit flow-field ( $\mathbf{V}$ ) applied on the surface. Informed by the literature on fluid simulation [7] and level-sets [44], deformation of a surface with such a flow field depends on the geometry representation.

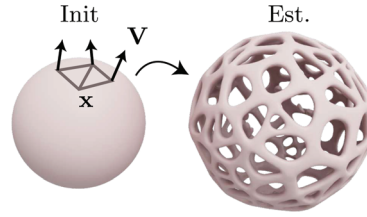
The *Lagrangian* representation involves tracking the surface explicitly as a set of points ( $\mathbf{x}$ ) and connections (like triangles). The point-set is discrete and the connectivity is *static*, which keeps the optimization relatively simple; we can separately integrate the field at each point (update vertices  $\mathbf{x} \rightarrow \mathbf{x}'$ ). But optimization of the resolution of the surface is non-trivial and can also get unwieldy for problems which involve surfaces with unknown topology.



Alternatively, *Eulerian* descriptions can be used. Each point in space has an object-property  $\phi$  associated with it, like the distance from the surface or its occupancy inside the enclosed volume. The surface here is implicitly defined with *dynamic* connectivity; one can smoothly vary the topology during optimization. Canonically,  $\phi$  is defined only on a discrete voxel-grid and needs to be interpolated for points off the grid. Here, making instantaneous updates to the surface is more involved as it requires changing  $\phi$  values for a large set of points. A neural implicit is a continuous variant of an Eulerian representation. Applying a flow field to such functions is non-trivial as updates are required in the parameter ( $\theta$ ) space as opposed to directly updating  $\phi$  to  $\phi'$ .

We propose a parametric level-set evolution method (Sect. 4) which propagates neural implicit surfaces according to an explicitly generated flow field. Our method comprises of three repeating steps, 1) A non-differentiable surface extraction method like Marching Cubes [31] or Sphere Tracing [21] is used to obtain a Lagrangian representation corresponding to a neural implicit, 2) A mesh-based algorithm is used to derive a flow field on the explicit surface (Sect. 4.1), and 3) A corresponding Eulerian flow field is used to evolve the implicit geometry (Sect. 4.2).

Previous methods [41, 49, 51] approach the problem of using mesh-based energy functions on Eulerian surfaces with the idea of differentiable extraction of Lagrangian representations. We show these methods are not in accordance with the level-set theory [45] (Sect. 5). The discussion also yields a more general proof (Sect. 5.1) for differentiable surface extraction from level set functions with arbitrary gradient norms ( $|\nabla\phi| \neq 1$ ). Our method is more formally connected to the theory and we validate it with experimental observations made on three diverse problem settings, 1) Curvature-based deformation (Sect. 6.1), where we demonstrate more accurate surface smoothing and mean-curvature flow than previous methods [49, 67], 2) Inverse rendering of geometry (Sect. 6.2), where we show accurate recovery from multi-view images for high-genus shapes without object masks as in [41, 69] (example in Fig. 1), and 3) User-defined shape editing (Sect. 6.3), where the implicit surface is deformed  $60\times$  faster than previous work [67].



**Fig. 1.** Inverse-rendering recovery.

## 2 Related Work

Coordinate-based MLPs have become a popular choice as function approximators for signals like images, videos and shapes [13, 46, 53, 56, 60]. Our work focuses on using MLPs for approximating implicit geometry. Such representations are compact, continuous and differentiable. These advantages are well suited with gradient-descent based optimization and learning problems. Recent developments in generative modeling of shapes [11, 35, 46, 52], 3D consistent image synthesis [12, 66], 3D reconstruction [5, 57, 72] and inverse rendering [25, 43, 68, 69], all rely on representing geometry using MLPs. For a more detailed discussion on recent work regarding coordinate-based representations refer to the survey by Xie *et al.* [65], and for inverse rendering the survey by Tewari *et al.* [63].

There is also a rich literature on geometry processing [8–10, 54, 55] and inverse rendering [4, 18, 32, 40] for explicit surface representations. Yang *et al.* [67] introduce some of the geometry processing ideas to neural implicit surfaces, but the method could be inaccurate (Sect. 6.1) and slow (Sect. 6.3).

For inverse rendering applications, differentiable renderers for triangle meshes are used for gradient-based optimization. Physics-based renderers differentiate through a light-simulation process [6, 27, 42, 71] and provide accurate gradients. Alternatively, differentiable rasterization [26, 30, 48] can be used for high-performance gradient computation, but only for single-bounce rendering models. However, optimizing triangle meshes with gradient-descent is non-trivial. Careful design of the optimization method [18, 40] and error functions [32] is required for robust optimization. To circumvent some of these issues, IDR [69] and DVR [41] use implicit surface representations like SDFs and occupancy functions [36] parameterized with MLPs. These methods mitigate some of the topological restrictions, but are not physics-based and are not in accordance with the

level-set theory (Sect. 5.2). We propose an inverse rendering method which uses explicit differentiable renderers for parametrically defined implicit surfaces. The proposed method is not as sensitive to initialization as explicit methods [18, 40] are, does not require an object mask like implicit methods [41, 69] do, and maintains the ability to vary topology during optimization.

Our method uses the level-set theory [45] as the foundation for optimizing and deforming parametric implicit surfaces. Previous methods [3, 39, 64] for the applications discussed in this work apply to non-parametric level sets. Perhaps the most related works to our method are MeshSDF [49] and RtS [15]. Compared to MeshSDF, our approach is more formally connected to the theory of level-sets (Sect. 4), applies to all parametric functions (Sect. 5.1), and works for a more diverse set of optimization problems like shape editing (Sect. 6.1, 6.3) and inverse rendering (Sect. 6.2) – deviating from experimental observations made by Remelli *et al.* [49] on learning-based settings. Compared to RtS [15], we show geometry processing applications along with theoretical parallels (Sect. 5) between parametric level-set methods like MeshSDF [49], DVR [41], IDR [69] and the classical theory [44]. Our inverse rendering method is shown to work (Sect. 6.2) for a set of high-genus shapes with a genus-0 initialization, in contrast to object-pose optimization or small genus changes shown in [15]. Recent work by Munkberg *et al.* [38] and Hasselgren *et al.* [22] also show promise in using explicit differentiable renderers with implicit geometry for inverse problems.

### 3 Background

Consider a closed surface of arbitrary topology  $\partial\Omega$  evolving with respect to time  $t$ . We define a Lagrangian representation of this surface with a finite set of  $k$  points in  $\mathbb{R}^3$  as  $\partial\Omega_L = \{\mathbf{x}_i \mid \mathbf{x}_i \sim \partial\Omega; \forall i \in \{1, 2, 3, \dots, k\}\}$ . This point-set can be viewed as a triangle mesh if an additional set of connections between the points is provided, and as a point cloud otherwise. Implicitly this surface can also be represented with a family of level-sets  $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}$ , the zero iso-contour of which represents the surface  $\partial\Omega_E = \{\mathbf{x} \mid \phi(\mathbf{x}) = 0\}$ .  $\phi$  can be arbitrarily chosen, but a canonical choice is a signed-distance function (SDF) which satisfies:

$$\phi(\mathbf{x}) = (\pm) \min_{\mathbf{x}_C \in \partial\Omega} \{\|\mathbf{x} - \mathbf{x}_C\|_2\}, \quad (1)$$

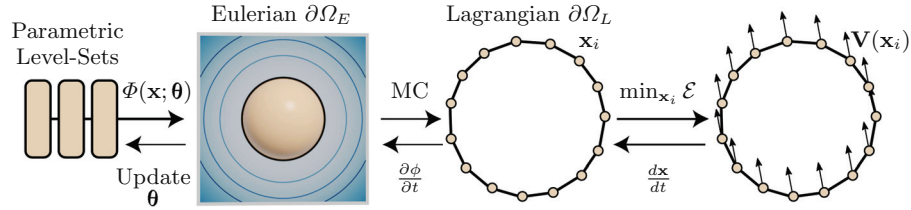
where  $\mathbf{x}_C$  is the closest point on the surface to  $\mathbf{x}$  and the sign of  $\phi(\mathbf{x})$  denotes whether  $\mathbf{x}$  is enclosed (−) within the shape or not (+).

*Parameterizing  $\phi$ .* Analytically defining  $\phi$  for simple and regular shapes is relatively straightforward [47], but is infeasible for most objects. Recent work on 3D reconstruction [41, 69] and generative shape modeling [13, 46] suggests parameterizing  $\phi$  using a multi-layer perceptron (MLP) with  $\theta$  as its parameters. The networks are optimized by minimizing an energy function comprised of a distance term [46] and a gradient term [19] enforcing  $|\nabla\phi|$  to be 1. We use SIREN [53] as the parametric function of choice, although our method is agnostic to the

network parameterization. The network acts approximately as an SDF at the rest state ( $t = 0$ ), but may not retain the SDF property (Eq. 1) as the surface evolves. We denote this surface as  $\partial\Omega_E(\theta) = \{\mathbf{x} \mid \Phi(\mathbf{x}; \theta) = 0\}$ , where  $\Phi$  is parameterized with  $\theta$  as the weights and biases of the network. For clarity, we use  $\Phi$  for parametric level-sets and  $\phi$  for non-parametric.

## 4 Method

We begin the discussion by first characterizing the deformation of surfaces into Lagrangian (Sect. 4.1) and Eulerian (Sect. 4.2) settings. We show that gradient descent on energy functions defined for triangle meshes can be viewed as surface deformation under the dynamics of a flow field  $\mathbf{V}$ , which is *discretely* defined only on the surface points  $\partial\Omega_L$ . We can use this flow field to deform a *continuous* surface representation using the level-set equation. We extend the level-set equation to the case of parametric level-sets  $\Phi$  which enables us to use loss functions defined on triangle meshes to deform MLP-defined level-sets.



**Fig. 2. Method Overview.** We present a level-set method to evolve neural representations of implicit surfaces. Using Marching Cubes (MC) [31], a Lagrangian surface  $\partial\Omega_L$  is extracted from an Eulerian representation  $\partial\Omega_E$  encoded in the network parameters  $\theta$ . An energy function  $\mathcal{E}$  is defined on  $\partial\Omega_L$  which is minimized using gradient-descent. The gradients of the optimizer together act as a flow-field  $\mathbf{V}$  on the surface points  $\mathbf{x}$ , which is used to evolve the non-parametric  $\phi$  using the level-set equation. The values of  $\phi$  on the surface act as references to update the parameters  $\theta$  of the network.

### 4.1 Lagrangian Deformation

As mentioned earlier, in the Lagrangian setting, the surface is defined with a finite set of points  $\partial\Omega_L$ . A variety of methods in geometry processing and computer vision define an energy function  $\mathcal{E}$  that is minimized to make instantaneous updates to  $\partial\Omega_L$ . Some recent examples include optimizing point-clouds for view-synthesis [1, 50] and triangle-meshes for inverse-rendering of geometry [18, 40]. The surface is updated using spatial gradients  $\frac{\partial \mathcal{E}}{\partial \mathbf{x}}$ , which is well studied in numerical analysis [58] and optimization methods [14]. Through the lens of physics, these gradients induce an instantaneous flow field  $\mathbf{V}(\mathbf{x})$ , which can be used to

evolve the surface by integrating the following ordinary differential equation (ODE):

$$\frac{d\mathbf{x}}{dt} = -\frac{\partial \mathcal{E}}{\partial \mathbf{x}} \rightarrow \mathbf{V}(\mathbf{x}). \quad \triangleleft \text{Lagrangian Deformation} \quad (2)$$

Numerically, this can be done using forward-Euler steps  $\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{V}^t(\mathbf{x})$ . This is easy to accomplish if the connectivity of the points remains static. More sophisticated integration schemes can also be used [58]. Here, in case of optimization problems solved using gradient descent,  $\Delta t$  is equivalent to the learning rate. Several works in shape deformation [20] and inverse rendering [40] can be subsumed by this ODE with different definitions for flow  $\mathbf{V}$  and time-step  $\Delta t$ . We next show that these readily available energy functions and optimization algorithms defined for explicit surfaces can also be used to optimize MLP-defined level-sets.

## 4.2 Eulerian Deformation

To avoid the topological complications associated with Lagrangian deformations, we can instead define a corresponding Eulerian deformation field. By definition, we know for points  $\mathbf{x} \in \partial\Omega$ ,  $\phi(\mathbf{x}) = 0$ . Using implicit differentiation:

$$\begin{aligned} \frac{d\phi(\mathbf{x})}{dt} &= \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \phi}{\partial t} + \nabla \phi \cdot \mathbf{V} = 0 \quad \triangleleft \text{From 2} \\ \iff \frac{\partial \phi}{\partial t} &= -\nabla \phi \cdot \mathbf{V}. \quad \triangleleft \text{Eulerian Deformation} \end{aligned} \quad (3)$$

This partial differential equation (PDE) is sometimes referred to as the level-set equation [45], the material derivative [7] or the G-equation [33]. We extend this PDE to obtain an evolution method for parametric level-sets  $\Phi$ . First, for each time step  $t$ , we extract a Lagrangian surface representation  $\partial\Omega_L^t$  from  $\Phi$  using MC [31]. Depending on the task at hand, an energy function  $\mathcal{E}$  (*e.g.*, photometric error for inverse rendering) is defined on  $\partial\Omega_L^t$ . Assuming  $\mathcal{E}$  is differentiable, we compute  $\frac{\partial \mathcal{E}}{\partial \mathbf{x}} = -\mathbf{V}^t(\mathbf{x})$  for each vertex  $\mathbf{x}_i \in \partial\Omega_L$ . With the flow field  $\mathbf{V}^t$ , we update the level-set function as we would in the non-parametric case using forward-Euler steps:

$$\phi^{t+1} = \Phi^t - \Delta t \nabla \Phi^t \cdot \mathbf{V}^t. \quad \triangleleft \text{From (3)} \quad (4)$$

The time step  $\Delta t$  here is a parameter which is dependent on the dynamics of the flow-field. If  $\mathbf{V}$  is highly non-linear, taking smaller steps (*i.e.*,  $\Delta t$  is small) is required, while for a simple field, larger values of  $\Delta t$  should suffice. For each time step, we take the values of non-parametric  $\phi^{t+1}$  as the reference and update the parameters of  $\Phi$  accordingly. This is achieved by minimizing the following objective:

$$\min_{\theta} J(\theta) = \frac{1}{|\partial\Omega_L|} \sum_{\mathbf{x} \in \partial\Omega_L} \|\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x}; \theta)\|^2, \quad (5)$$

using gradient descent. Since the surface updates are small for each time step, the number of descent steps required is in the order of  $10^2$ . This makes the method convenient for obtaining neural representations of deformed variants of the initial geometry. After each optimization routine, we again extract the Lagrangian surface using MC [31], which is subsequently fed into a mesh-based energy-minimization problem. An overview of our method is shown in Fig. 2. For each of the applications we show in Sect. 6, we define  $\mathbf{V}^t$  using a corresponding energy function, and minimize  $J$  for each time step.

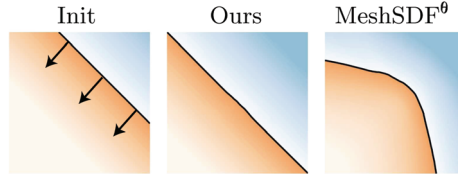
## 5 Theoretical Comparisons

The level-set method discussed in Sect. 4.2 subsumes two related works, 1) Differentiable iso-surface extraction (MeshSDF) [49], and 2) Differentiable rendering of implicit surfaces (DVR/IDR) [41, 69]. We first show that MeshSDF minimizes the level-set objective  $J$  defined in (5) with a single gradient-descent step. But the surface does not propagate in agreement with the level-set equation, as outlined in Sect. 5.1. We then show that these two seemingly disparate works (MeshSDF and DVR) are closely related in Sect. 5.2. We end the discussion with an explanation for how DVR deviates from the level-set equation.

### 5.1 Differentiable Iso-Surface Extraction

**Result 1.** *Differentiable Iso-Surface Extraction [49] takes a single gradient-descent step to minimize the level-set objective function  $J$  (Eq. 5).*

*Proof.* MeshSDF [49] defines a loss-function  $\mathcal{L}$  on a triangle mesh extracted using Marching Cubes [31] from an SDF parameterized with an MLP. They use an MLP  $\Phi(\mathbf{x}; \theta, \mathbf{z})$  conditioned on a latent-code  $\mathbf{z}$  characterizing the shape. Using  $\mathcal{L}$  they update the latent-code  $\mathbf{z}$ , which is different from our goal of updating  $\theta$  for an unconditional  $\Phi$ . To clarify this distinction, we use MeshSDF <sup>$\theta$</sup>  to denote our variant which updates  $\theta$ . To update the parameters of the MLP, we compute the following gradient using the chain-rule:



**Fig. 3. MeshSDF <sup>$\theta$</sup>  does not follow the level-set equation.** (Left) A planar surface defined implicitly with an MLP is influenced by a flow field in the direction of its normal. The motion attained using differentiable iso-surface extraction (Right) is inconsistent with the field. Our method (Center) propagates the front as expected.

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{\mathbf{x} \in \partial \Omega_L} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \Phi} \frac{\partial \Phi}{\partial \theta}, \quad (6)$$

where  $\mathbf{x}$  are the vertices on the mesh and  $\Phi$  is an SDF. The first and the third gradient terms on the right are computed using automatic differentiation. The second term  $\frac{\partial \mathbf{x}}{\partial \Phi}$  can be approximated as the inverted surface normal  $-\mathbf{n}(\mathbf{x}) = -\nabla_{\mathbf{x}}\Phi$  [49], when  $\Phi$  is an SDF. In the spirit of Lagrangian deformation (Eq. 2),  $-\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$  acts as an instantaneous flow field  $\mathbf{V}$  on the vertices  $\mathbf{x}$ . The parameters of the MLP are then updated as:

$$\theta \leftarrow \theta - \lambda \frac{\partial \mathcal{L}}{\partial \theta} = \theta - \lambda \sum_{\mathbf{x} \in \partial \Omega_L} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}, \quad (7)$$

where  $\lambda$  is the learning rate. Alternatively, we can also update  $\theta$  using the objective function defined in Eq. 5 using gradient descent:

$$\begin{aligned} \theta \leftarrow \theta - \lambda \frac{\partial J}{\partial \theta} &= \theta - \lambda \sum_{\mathbf{x} \in \partial \Omega_L} 2(\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x}; \theta)) \left( -\frac{\partial \Phi}{\partial \theta} \right) \\ &= \theta - \epsilon \sum_{\mathbf{x} \in \partial \Omega_L} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}, \quad \triangleleft \text{From (4)} \end{aligned} \quad (8)$$

where  $\epsilon$  is a constant. The last equivalency is valid when  $\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x}; \theta) = -\Delta t \mathbf{V} \cdot \nabla \Phi$  (Eq. 4), which is true only for the first gradient descent step. Subsequently, the value of  $\Phi(\mathbf{x}; \theta)$  changes as the parameters get updated. Comparing (7) and (8), we conclude that the optimization in MeshSDF<sup>θ</sup> has the effect of taking a single gradient-descent step to minimize  $J$ . Note that while the proof by Remelli *et al.* [49] assumes  $\Phi$  is an SDF, the second update equation (8) does not. It is valid for all level-set functions, with no restrictions on the values of the gradient-norm ( $|\nabla \Phi|$ ) and is also valid for occupancy functions [36].  $\square$

However, by taking a single step to update  $\theta$ , the surface does not propagate in agreement with the level-set Eq. (3). This is problematic for applications which require the surface to move as intended by the flow-field. An example application is shown in Sect. 6.1. We also illustrate this with a toy example in Fig. 3 where a planar surface propagates in the direction of its normal. A more formal discussion is in Result 2.

**Result 2.** *Differentiable iso-surface extraction [49] does not propagate the surface-front as dictated by the flow field.*

*Proof.* We show this with an example flow field. Consider the surface-front propagating with a constant speed  $\beta$  in the direction of the normal. The corresponding Eulerian deformation is constant across the surface:

$$\frac{\partial \phi}{\partial t} = -|\nabla \phi| \beta = -\beta. \quad \triangleleft \text{Assuming } \phi \text{ is an SDF} \quad (9)$$

With the same flow field, we can estimate the instantaneous change in  $\Phi$  (parametric) for MeshSDF<sup>θ</sup> as:

$$\frac{\partial \Phi}{\partial t} = \frac{\partial \Phi}{\partial \theta} \frac{\partial \theta}{\partial t} = \frac{\partial \Phi}{\partial \theta} \sum_{\mathbf{x} \in \partial \Omega_L} -\beta \frac{\partial \Phi}{\partial \theta} = \frac{\partial \Phi}{\partial \theta} B. \quad \triangleleft \text{From (7) and (9)} \quad (10)$$

The term  $B$  on the right is constant for every point  $\mathbf{x}$  on the surface. The gradient  $\frac{\partial \Phi}{\partial \theta}$  is dependent on the position where it is evaluated and hence front-propagation  $\frac{\partial \Phi}{\partial t}$  is not constant. On the contrary, we minimize the objective function defined in (5) which ensures that the surface propagation is constant as in (9).  $\square$

For the implications of Result 1 and 2, and experimental comparisons, we defer the discussion to Sect. 6.1.

## 5.2 Differentiable Surface Rendering

An alternate way of extracting a Lagrangian surface  $\partial\Omega_L$  from  $\Phi$  is by computing ray-surface intersections using ray-marching or sphere-tracing [21]. If ray-marching is differentiable [24, 29], one can backpropagate gradients from error functions defined on  $\partial\Omega_L$  to the parameters  $\theta$  defining the implicit surface. Recent developments in inverse rendering [36, 69] rely on this idea. The explicit surface extracted using ray-marching differs from the one obtained using marching-cubes in two ways, 1) Ray-marching does not extract the connectivity (*e.g.*, triangle faces) among the intersection points. This restricts the usage of loss functions which rely on attributes like the edge-length or differential operators like the Laplacian. 2) The intersection points depend on the camera attributes. The resolution of the image plane affects the density of points and the viewing direction determines the visibility. As a result,  $\partial\Omega_L$  obtained using ray-marching could be sparser than the one obtained marching cubes. Furthermore, as we show in Result 3, by using differentiable ray-marching the parameters get updated exactly as when differentiable iso-surface extraction is used—although with a less favorable Lagrangian representation (sparser and no connectivity). This is in addition to the computational disadvantage associated with ray-marching. We also formally show in Result 4 that surface evolution with differentiable ray-marching is in disagreement with level-set theory for tangential flows.

**Result 3.** *Surface evolution using differentiable ray-marching of parametric implicit surfaces [36, 69] is the same as using differentiable iso-surface extraction [49] when the viewing direction  $\mathbf{v}_u$  is parallel to the normal  $\mathbf{n}$  at the intersection point  $\mathbf{x}_u$ . The parameters  $\theta$  for the level-set function  $\Phi$  are updated as:*

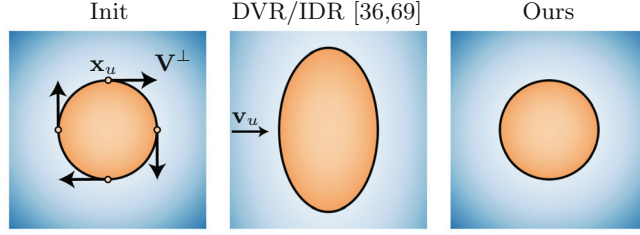
$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}, \quad (11)$$

where  $\mathbf{x}_u$  are the visible points and  $\mathbf{V}$  is the flow field. Comparing (11) and (7), the gradient-descent step is the same. As in the case of MeshSDF<sup>θ</sup>, here the surface is evolved with a single step in the parameter space. We provide a more detailed proof for (11) in the Appendix.

**Result 4.** *Differentiable ray-marching of parametric implicit surfaces [36, 69] disagrees with the level-set equation for tangential components  $\mathbf{V}^\perp$  of the flow field  $\mathbf{V}$ . The change in parameters  $\Delta\theta$  is:*

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^\perp| \tan(\arccos(\nabla\Phi \cdot \mathbf{v}_u)) \frac{\partial\Phi}{\partial\theta} \neq 0, \quad (12)$$

which could be non-zero. A detailed proof is in the Appendix. Referring to Eq. 3, for tangential flows the surface should not undergo any deformation, i.e.  $\frac{\partial\phi}{\partial t} = 0$ . However, since  $\theta$  gets updated as in (12), the surface *does* deform. We instead minimize the objective function  $J$  (Eq. 5) which is 0 for a tangential field. We show an example of tangential deformation in



**Fig. 4. Tangential flow fields may deform surfaces when DVR/IDR is used for surface extraction.** (Left) A parametric Eulerian circle undergoes tangential deformation  $\mathbf{V}^\perp$  at surface points  $\mathbf{x}_u$ . (Middle) Using differentiable surface rendering, the surface deforms incorrectly. (Right) Our method agrees with the level-set equation and the resultant deformation is the identity.

Fig. 4. Experimental comparisons in Sect. 6.2 validate that deforming  $\Phi$  accurately is critical for applications like inverse rendering.

## 6 Applications

We focus on validating the proposed theory with computer graphics models in three different settings, 1) Curvature-based deformations (Sect. 6.1), which can be used to smooth/sharpen features and apply curvature defined flows on implicit surfaces, 2) Inverse rendering of geometry (Sect. 6.2), where a differentiable renderer for explicit geometry can be used to evolve implicit surfaces, and 3) User-defined deformations (Sect. 6.3), where a user can specify alterations for a given object.

### 6.1 Curvature-based Deformation

To apply surface smoothing on parametric implicit surfaces, we first define a corresponding explicit force field. For the extracted Lagrangian surface  $\partial\Omega_L$  at each time step, we minimize the Dirichlet-energy functional on the surface. In the continuous setting, this is defined as:

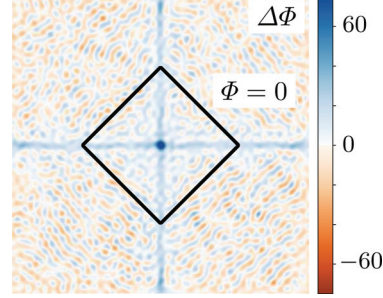
$$\mathcal{E}(\partial\Omega) = \int_{\partial\Omega} \|\nabla \mathbf{x}\|^2 d\mathbf{x}. \quad (13)$$

Minimizing  $\mathcal{E}$  can be shown [8] to induce the following explicit flow-field on the surface:

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{V}(\mathbf{x}) = \lambda \Delta \mathbf{x} = -2\lambda \kappa(\mathbf{x}) \mathbf{n}(\mathbf{x}), \quad (14)$$

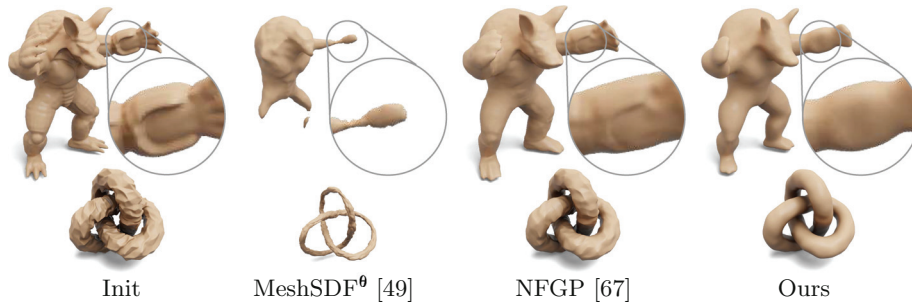
where  $\lambda$  is a scalar diffusion coefficient,  $\Delta$  is the Laplace-Beltrami operator and  $\mathbf{n}$  is the normal.  $\kappa$  is the mean-curvature, which for an implicit surface is defined as the divergence of the normalized gradient of  $\phi$  (*i.e.*,  $\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$ ) [44]. It is equivalent to computing the laplacian  $\Delta\Phi$  of the MLP using automatic differentiation. In Fig. 5 we show that such an estimation of the laplacian is noisy—significant magnitudes are observed even for surfaces with zero curvature.<sup>1</sup> Instead of using (14) as it is, which requires estimating  $\Delta\Phi$ , we approximate  $\mathbf{V} = \lambda \Delta \mathbf{x} \approx \lambda \mathbf{L} \mathbf{x}$ ; where  $\mathbf{L}$  is the discrete Laplacian we can compute using the Lagrangian surface. Note that this is feasible only because of the hybrid (Eulerian+Lagrangian) nature of our method. We use the flow-field to update  $\Phi$  using the method outlined in Sect. 4.2. Figure 6 shows qualitative comparisons for smoothing applied on two surfaces. We show a comparison with a method which applies deformation using the continuous Laplacian ( $\Delta\Phi$ ) (NFGP) [67]. When  $\mathcal{E}$  is minimized using MeshSDF<sup>θ</sup> [49], the deformation is not curvature based and high-frequency features are retained during the evolution.

Equation (14) is referred to as mean-curvature flow [17]. Since our method deforms the surface in accordance with the flow-field, we can use (14) to apply mean-curvature flow on a parametrically defined  $\phi$ . Yang *et al.* [67] minimize an objective function which is handcrafted for a specific level-of-smoothness. Applying curvature-based flow is infeasible with their method since it would require a new optimization objective for each level-of-smoothness. As MeshSDF<sup>θ</sup> [49] does not evolve the surface according to the level-set equation, the flow obtained with it is incorrect. We show an example flow on a genus-0 surface in Fig. 7.



**Fig. 5. The laplacian  $\Delta\Phi$  of an MLP-defined level-set function is noisy.** We show the mean-curvature values for a parametric level-set function of a square. Large values are observed for a zero-curvature surface.

<sup>1</sup> This might be due to the unconstrained Lipschitz constants of MLPs [28].

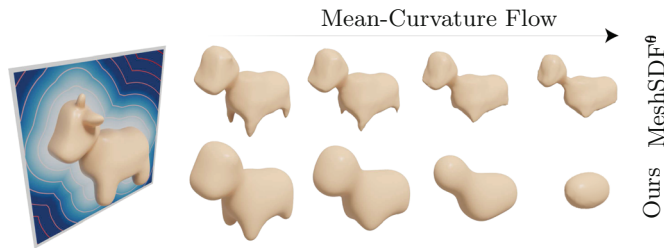


**Fig. 6. Surface smoothing on parametric level-sets.** We apply surface smoothing on an MLP-defined implicit surface by minimizing Dirichlet energy on the corresponding explicit surface. We use a discrete Laplacian to define a flow-field on the surface; NFGP [67] uses its continuous counterpart and preserves too many high-frequency details. MeshSDF<sup>θ</sup> [49] fails to smoothen the surface.

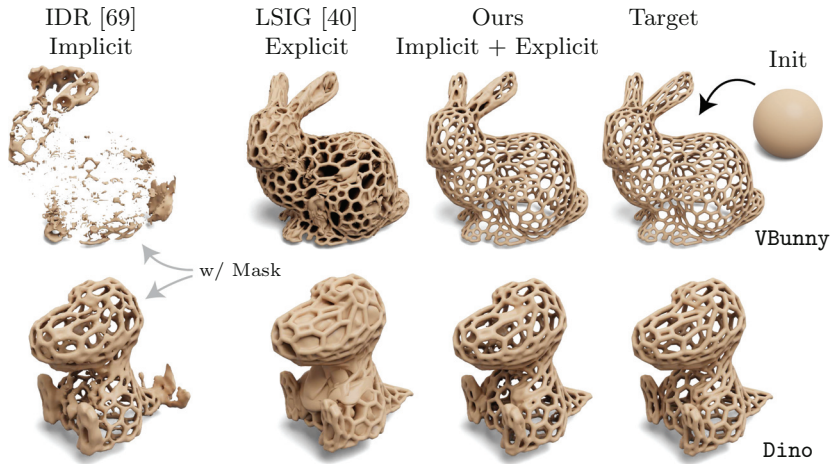
## 6.2 Inverse Rendering of Geometry

We propose an inverse-rendering method which uses a differentiable renderer designed for triangle meshes to optimize geometry defined using parametric level-sets. As in the case of recent methods [4, 41, 69], we use an analysis-by-synthesis approach. A photometric error comparing the captured and the rendered images is minimized using gradient descent. The gradients of the error function are used to define an explicit flow-field. A corresponding Eulerian deformation field is obtained to evolve the level-set function  $\Phi$ . As a result we can take large steps in inverse rendering of geometry with unconstrained topology and guarantees on mesh quality. The resulting optimization is robust and does not require an object-mask as in the (unlike [41, 69]).

Our single-bounce forward rendering model uses a collocated camera and point-light, with a known diffuse-Phong BRDF. Although we choose Nvd-iffrastr [26] as the differentiable rasterizer in our method, in theory it can be

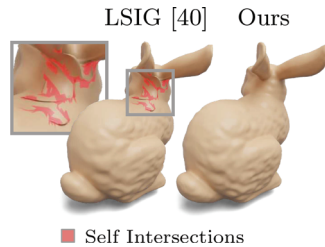


**Fig. 7.** We use an explicit mean-curvature flow-field to deform a parametrically defined implicit surface. When the same flow-field is used with MeshSDF, the deformation is not curvature-based. A genus-0 surface morphs into a sphere using our method while MeshSDF retains high-curvature regions.



**Fig. 8. Inverse rendering of high-genus shapes.** A spherical surface is used for initialization. IDR [69], which uses differentiable rendering of implicit surfaces does not recover finer details. LSIG [40] uses a triangle-mesh and restricts the topology post initialization. Using an explicit differentiable renderer to optimize implicit geometry, our method can change topology during optimization and recover fine-details. Note that IDR requires an object mask and a neural renderer.


swapped with any other differentiable renderer. Starting from an initial estimate  $\Phi$ , for each time-step  $t$  we first extract the triangle mesh  $\partial\Omega_L^t$  and minimize a photometric error  $\mathcal{E}$ . We use the gradients of  $\mathcal{E}$  to define the flow-field as  $\mathbf{V}^t(\mathbf{x}_i) = -\frac{\partial\mathcal{E}}{\partial\mathbf{x}_i} - \lambda\mathbf{L}\mathbf{x}_i$ , where  $\mathbf{L}\mathbf{x}_i$  is used for smooth evolution. Taking a single descent step for  $\theta$  is sufficient since the evolution does not need to follow a specific trajectory.



geometry, but the optimized meshes can have self intersections as shown on the left. It struggles with high-genus shapes as the mesh connectivity remains static throughout the optimization routine. Even with correct topology at initialization, the recovery is erroneous. Comparisons with IDR [69] are in Fig. 9.

We evaluate the recovery on a diverse set of shapes, each of which is rendered from 100 random viewpoints. We use IDR [69] and LSIG [40] as baselines. We test IDR in three settings, 1) w/o Mask, 2) With known Phong shading, and 3) Using the Neural renderer in [69]. Quantitative comparisons are in Table 1 and qualitative comparisons are in Fig. 8. For a genus-0 shape (Bunny) LSIG [40] is able to recover accurate

**Table 1. Quantitative evaluation on inverse rendering of geometry.** An initial sphere is optimized to a diverse set of shapes from multi-view images. Chamfer distance is reported for geometric consistency and PSNR is reported to evaluate the visual appearance of optimized geometry. IDR [69] w/o Mask does not converge for any of the shapes. Methods marked with † require object masks. LSIG [40] works well for a genus-0 shape (Bunny) but struggles with high-genus shapes. Shapes recovered using our method are shown on the top.



Method	Bunny	Genus6	Rind	VSphere	Dino	VBunny	Kangaroo	
†IDR - Phong [69]	9.67	3.71	10.24	16.93	3.71	14.80	2.77	
†IDR - Neural [69]	9.84	1.35	0.21	0.16	2.07	9.11	5.43	Chamfer
LSIG [40]	0.06	2.85	3.94	4.78	2.09	4.66	1.80	$\times 10^{-3}$
Ours	0.18	0.12	5.56	3.71	1.25	0.10	1.62	↓ better
†IDR - Phong [69]	21.52	18.84	15.89	18.28	20.01	17.27	21.67	
†IDR - Neural [69]	23.10	28.70	26.24	25.70	22.49	16.62	21.74	PSNR
LSIG [40]	38.51	25.67	22.81	24.06	25.11	21.77	25.52	dB
Ours	38.86	32.94	28.46	30.50	28.16	29.62	26.48	↑ better

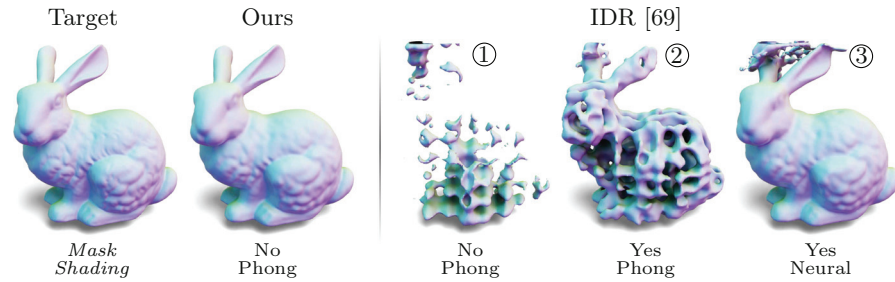
### 6.3 User-Defined Shape Editing



**Fig. 10.** User-defined editing on parametric level-sets.

We demonstrate deformation operations on parametric level-sets using constraints defined by a user. The problem setup is in line with the extensive literature [10, 54, 55, 70] on shape editing for triangle meshes. At  $t = 0$ , we first extract a mesh from a neural implicit surface using MC [31]. A user can specify handle regions on this surface to either rotate, translate or freeze parts of the shape along with their target locations. This generates a sparse deformation flow-field on the surface which we *densify* by minimizing a thin-shell energy function that penalizes local stretching and bending on the surface [62]. More details are in the Appendix. Estimating the flow-field requires solving a linear system  $-k_S \Delta \mathbf{V} + k_B \Delta^2 \mathbf{V} = 0$ , where  $\Delta$  and  $\Delta^2$  are the laplacian and bi-laplacian operators.  $k_S$  and  $k_B$  are weighting terms for stretching and bending respectively. Additional constraints which adhere to user specifications are also added to the linear system [9]. With the obtained flow-field, we update the parameters of the level-set function such that the surface propagation is as intended. We also use gradient regularization as in [19]. An example deformation is shown in Fig. 10.

We demonstrate deformation operations on parametric level-sets using constraints defined by a user. The problem setup is in line with the extensive literature [10, 54, 55, 70] on shape editing for triangle meshes. At  $t = 0$ , we first extract a mesh from a neural implicit surface using MC [31]. A user can specify handle regions on this surface to either rotate, translate



**Fig. 9. Qualitative comparison for inverse rendering using implicit representations.** We evaluate IDR in three different settings. ① Without object-mask supervision it fails to converge to a reasonable geometry. ② With a known-reflectance model (Phong) the silhouette of the object is recovered but without any details. ③ It requires a rendering network (unknown reflectance) and an object-mask for good convergence—both of which not required by our method.

## 7 Discussion



**Fig. 11. Joint recovery of geometry, complex material and lighting.**

Our work formulates a level-set evolution method for parametrically defined implicit surfaces. It does not require surface extraction to be differentiable and can be used to apply mesh algorithms to neural implicit surfaces. We expect the proposed method to be particularly useful for inverse problems. We showcase one example of joint recovery of geometry, material and lighting from multi-view images in Fig. 11, where we use our surface evolution method along with components from [38]. Although the surface deformation is as dictated by the flow field, the corresponding implicit function may not retain gradient characteristics during evolution. This could become a pertinent problem for algorithms like sphere tracing [21] which require reliable distance queries, and is an interesting avenue for future research. We hope this work encourages further inquiry into recent work on geometry optimization by drawing connections to methods in computer vision and graphics developed in the pre-deep-learning era.

**Acknowledgements.** This work was supported in part by NSF CAREER 1751365, NSF IIS 2110409, ONR grant N000142012529, NSF Chase-CI grant 1730158, Adobe, Google, an Amazon Research Award, the Ronald L. Graham Chair and UC San Diego Center for Visual Computing. We thank Ceh Jan and 3dmixers users [roman.hegglin](#) and [PhormaSolutions](#) for the 3D models. We thank the anonymous reviewers for helpful comments and discussions.

## References

1. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics (2020)
2. Alldieck, T., Xu, H., Sminchisescu, C.: imghum: implicit generative models of 3D human shape and articulated pose. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5461–5470 (2021)
3. Ambrosio, L., Soner, H.M.: Level set approach to mean curvature flow in arbitrary codimension. *J. Differ. Geom.* **43**(4), 693–737 (1996)
4. Azinović, D., Li, T.M., Kaplanyan, A., Nießner, M.: Inverse path tracing for joint material and lighting estimation. In: Proceedings of the Computer Vision and Pattern Recognition (CVPR). IEEE (2019)
5. Azinović, D., Martin-Brualla, R., Goldman, D.B., Nießner, M., Thies, J.: Neural RGB-D surface reconstruction. arXiv preprint [arXiv:2104.04532](https://arxiv.org/abs/2104.04532) (2021)
6. Bangaru, S., Li, T.M., Durand, F.: Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* **39**(6), 245:1–245:18 (2020)
7. Batchelor, C.K., Batchelor, G.: An Introduction to Fluid Dynamics. Cambridge University Press, Cambridge (2000)
8. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B.: Polygon Mesh Processing. CRC Press, New York (2010)
9. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. *IEEE Trans. Visual. Comput. Graphics* **14**(1), 213–230 (2007)
10. Botsch, M., Sumner, R., Pauly, M., Gross, M.: Deformation transfer for detail-preserving surface editing. In: Vision, Modeling & Visualization, pp. 357–364. Cite-seer (2006)
11. Chabra, R., et al.: Deep local shapes: learning local SDF priors for detailed 3D reconstruction. arXiv preprint [arXiv:2003.10983](https://arxiv.org/abs/2003.10983) (2020)
12. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-GAN: periodic implicit generative adversarial networks for 3d-aware image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5799–5809 (2021)
13. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
14. Chong, E.K., Zak, S.H.: An Introduction to Optimization. Wiley, Hoboken (2004)
15. Cole, F., Genova, K., Sud, A., Vlasic, D., Zhang, Z.: Differentiable surface rendering via non-differentiable sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6088–6097 (2021)
16. Davies, T., Nowrouzezahrai, D., Jacobson, A.: On the effectiveness of weight-encoded neural implicit 3D shapes. arXiv preprint [arXiv:2009.09808](https://arxiv.org/abs/2009.09808) (2020)
17. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 317–324 (1999)
18. Goel, P., Cohen, L., Guesman, J., Thamizharasan, V., Tompkin, J., Ritchie, D.: Shape from tracing: towards reconstructing 3D object geometry and svbrdf material from images via differentiable path tracing. In: 2020 International Conference on 3D Vision (3DV), pp. 1186–1195. IEEE (2020)
19. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: Proceedings of Machine Learning and Systems 2020 (2020)

20. Gupta, K., Chandraker, M.: Neural mesh flow: 3D manifold mesh generation via diffeomorphic flows. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 1747–1758. Curran Associates, Inc. (2020). <https://proceedings.neurips.cc/paper/2020/file/1349b36b01e0e804a6c2909a6d0ec72a-Paper.pdf>
21. Hart, J.C.: Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *Visual Comput.* **12**(10), 527–545 (1996)
22. Hasselgren, J., Hofmann, N., Munkberg, J.: Shape, light & material decomposition from images using monte carlo rendering and denoising. *arXiv preprint arXiv:2206.03380* (2022)
23. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: SDFDIFF: differentiable rendering of signed distance fields for 3d shape optimization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1251–1261 (2020)
24. Jiang, Y., Ji, D., Han, Z., Zwicker, M.: SDFDIFF: differentiable rendering of signed distance fields for 3D shape optimization. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) June 2020*
25. Kellnhofer, P., Jebe, L.C., Jones, A., Spicer, R., Pulli, K., Wetzstein, G.: Neural lumigraph rendering. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4287–4297 (2021)
26. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. *ACM Trans. Graphics* **39**(6), 1–14 (2020)
27. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* **37**(6), 222:1–222:11 (2018)
28. Liu, H.T.D., Williams, F., Jacobson, A., Fidler, S., Litany, O.: Learning smooth neural functions via lipschitz regularization. *arXiv preprint arXiv:2202.08345* (2022)
29. Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., Cui, Z.: Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020)
30. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: a differentiable renderer for image-based 3D reasoning. In: *The IEEE International Conference on Computer Vision (ICCV)*, October 2019
31. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. *ACM SIGGRAPH Comput. Graphics* **21**(4), 163–169 (1987)
32. Luan, F., Zhao, S., Bala, K., Dong, Z.: Unified shape and SVBRDF recovery using differentiable monte Carlo rendering. *ArXiv* (2021)
33. Markstein, G.H.: *Nonsteady flame propagation: AGARDograph*. Elsevier (2014)
34. Martel, J.N.P., Lindell, D.B., Lin, C.Z., Chan, E.R., Monteiro, M., Wetzstein, G.: Acorn: adaptive coordinate networks for neural scene representation. *ACM Trans. Graph. (SIGGRAPH)* **40**(4) (2021)
35. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14214–14223, October 2021
36. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: learning 3D reconstruction in function space. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019)

37. Mu, J., Qiu, W., Kortylewski, A., Yuille, A., Vasconcelos, N., Wang, X.: A-SDF: learning disentangled signed distance functions for articulated shape representation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13001–13011 (2021)
38. Munkberg, J., et al.: Extracting Triangular 3D Models, Materials, and Lighting From Images. [arXiv:2111.12503](https://arxiv.org/abs/2111.12503) (2021)
39. Museth, K., Breen, D.E., Whitaker, R.T., Barr, A.H.: Level set surface editing operators. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 330–338 (2002)
40. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* **40**(6), December 2021. <https://doi.org/10.1145/3478513.3480501>, <https://rgl.epfl.ch/publications/Nicolet2021Large>
41. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: learning implicit 3D representations without 3d supervision. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020
42. Nimier-David, M., Vicini, D., Zeltner, T., Jakob, W.: Mitsuba 2: a retargetable forward and inverse renderer. *Trans. Graphics (Proc. SIGGRAPH Asia)* **38**(6), December 2019. <https://doi.org/10.1145/3355089.3356498>
43. Oechsle, M., Peng, S., Geiger, A.: UNISURF: unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5589–5599 (2021)
44. Osher, S., Fedkiw, R.P.: Level set methods: an overview and some recent results. *J. Comput. Phys.* **169**(2), 463–502 (2001)
45. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics* **79**, 12–49 (1988)
46. 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
47. Quilez, I.: <https://iquilezles.org/www/articles/distfunctions/distfunctions.htm>
48. Ravi, N., et al.: Accelerating 3d deep learning with pytorch3d. <http://arxiv.org/abs/2007.08501> (2020)
49. Remelli, E., et al.: MeshSDF: differentiable ISO-surface extraction. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 22468–22478. Curran Associates, Inc. (2020). <https://proceedings.neurips.cc/paper/2020/file/fe40fb944ee700392ed51bfe84dd4e3d-Paper.pdf>
50. Rückert, D., Franke, L., Stamminger, M.: ADOP: approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635* (2021)
51. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021)
52. Sitzmann, V., Chan, E.R., Tucker, R., Snavely, N., Wetzstein, G.: MetaSDF: meta-learning signed distance functions. In: *Proceedings of the NeurIPS* (2020)
53. Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: *arXiv* (2020)
54. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: *Symposium on Geometry Processing*, vol. 4, pp. 109–116 (2007)

55. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry Processing, pp. 175–184 (2004)
56. Stanley, K.O.: Compositional pattern producing networks: a novel abstraction of development. *Genetic Program. Evol. Mach.* **8**(2), 131–162 (2007)
57. Sucar, E., Liu, S., Ortiz, J., Davison, A.J.: imap: implicit mapping and positioning in real-time. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6229–6238 (2021)
58. Süli, E., Mayers, D.: An Introduction to Numerical Analysis. An Introduction to Numerical Analysis, Cambridge University Press (2003). <https://books.google.com/books?id=hj9weaqJTbQC>
59. Takikawa, T., et al.: Neural geometric level of detail: real-time rendering with implicit 3D shapes (2021)
60. Tancik, M., et al.: Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint [arXiv:2006.10739](https://arxiv.org/abs/2006.10739) (2020)
61. Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, pp. 351–358 (1995)
62. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 205–214 (1987)
63. Tewari, A., et al.: Advances in neural rendering. arXiv preprint [arXiv:2111.05849](https://arxiv.org/abs/2111.05849) (2021)
64. Whitaker, R.T.: A level-set approach to 3D reconstruction from range data. *Int. J. Comput. Vision* **29**(3), 203–231 (1998)
65. Xie, Y., et al.: Neural fields in visual computing and beyond. arXiv preprint [arXiv:2111.11426](https://arxiv.org/abs/2111.11426) (2021)
66. Xu, Y., Peng, S., Yang, C., Shen, Y., Zhou, B.: 3D-aware image synthesis via learning structural and textural representations (2021)
67. Yang, G., Belongie, S., Hariharan, B., Koltun, V.: Geometry processing with neural fields. In: Thirty-Fifth Conference on Neural Information Processing Systems (2021)
68. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: Advances in Neural Information Processing Systems 34 (2021)
69. Yariv, L., et al.: Multiview neural surface reconstruction by disentangling geometry and appearance. In: Advances in Neural Information Processing Systems, vol. 33 (2020)
70. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. In: Computer Graphics Forum, vol. 24, pp. 601–609. Citeseer (2005)
71. Zhang, C., Miller, B., Yan, K., Gkioulekas, I., Zhao, S.: Path-space differentiable rendering. *ACM Trans. Graph.* **39**(4), 143:1–143:19 (2020)
72. Zhu, Z., et al.: Nice-slam: Neural implicit scalable encoding for slam. arXiv preprint [arXiv:2112.12130](https://arxiv.org/abs/2112.12130) (2021)