Neural Implicit Surfaces in Higher Dimension

Tiago Novello, Vinicius da Silva, Helio Lopes,
Guilherme Schardong, Luiz Schirmer, Luiz Velho

January 28, 2022

Abstract

This work investigates the use of neural networks admitting high-order derivatives for modeling dynamic variations of smooth implicit surfaces. For this purpose, it extends the representation of differentiable neural implicit surfaces to higher dimensions, which opens up mechanisms that allow to exploit geometric transformations in many settings, from animation and surface evolution to shape morphing and design galleries.

The problem is modeled by a *k-parameter family* of surfaces S_c, specified as a neural network function $f : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$, where S_c is the zero-level set of the implicit function $f(\cdot, c) : \mathbb{R}^3 \to \mathbb{R}$, with $c \in \mathbb{R}^k$, with variations induced by the control variable c. In that context, restricted to each coordinate of $\mathbb{R}^k$, the underlying representation is a neural homotopy which is the solution of a general *partial differential equation*.

1 Introduction

A *smooth neural implicit function*, $g : \mathbb{R}^3 \to \mathbb{R}$, is a neural network capable of representing smooth implicit surfaces. Since g is smooth, concepts from *differential geometry* can be used, for instance, during the training of its parameters [25].

In this work, we investigate the possibilities of extending the domain of neural implicit functions from $\mathbb{R}^3$ to higher dimensions, for example to *space-time* $\mathbb{R}^3 \times \mathbb{R}$. This would allow the animation of a given implicit surface over time to be encoded into a single smooth function with domain in $\mathbb{R}^3 \times \mathbb{R}$. Here, $p \in \mathbb{R}^3$ denotes a point in the space, and $t \in \mathbb{R}$ denotes the time (the use of $\mathbb{R}$ instead of $[0, \infty)$ is an *abuse of notation*).

In a more general context, we take the domain as $\mathbb{R}^3 \times \mathbb{R}^k$, where these $k \in \mathbb{N}$ extra dimensions are used to model variation directions of an implicit surface S. The vector $c \in \mathbb{R}^k$ corresponds to control variables that can be used to explore the configuration space. Specifically, let $f : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ be a smooth function represented by a neural network such that S is the zero-level set of $f(\cdot, 0)$, where 0 is the origin of $\mathbb{R}^k$. The *k-parameter family* of smooth functions $f(\cdot, c) : \mathbb{R}^3 \to \mathbb{R}$ with $c \in \mathbb{R}^k$, represents a family of neural implicit functions. The animation of S is given by the *k-parameter* family of implicit surfaces S_c defined as the zero-level set of $f(\cdot, c)$.

The only requirements are that the function f must be smooth (or at least C^3) and be represented by a neural network. In this work, we focus on the space-time domain $\mathbb{R}^3 \times \mathbb{R}$.

The contributions of our work can be summarized as follows:

- The extension of differentiable neural implicit surfaces to higher dimensions;
- The introduction of a shape transformation framework based on neural homotopies;
- The development of machine learning techniques with applications on animation, surface evolution and morphing.

2 Related works

The research topics related to our work include: warping and morphing with implicit surfaces; level set methods; implicit surface representations using neural networks; differential geometry; exterior calculus; partial differential equations and physics-based methods.

Warping and morphing has been an active area of research in Computer Graphics since the 1990's [14]. Many problems in this area, such as shape correspondence and topology changes, can be elegantly solved using implicit surfaces [34],[2]. Additionally, animation of deformable and soft objects can be effectively posed in the context of implicit representations [3],[12],[11].

On the other hand, the generalization of shape interpolation inspired the development of useful tools in the context of Computer Aided Design (CAD). For example, design galleries provide an intuitive interface to explore and navigate families of shapes [22], [35].

The emergence of A.I. Graphics, has introduced machine learning methods that made possible great advances to reformulate and solve classical problems, creating new applications. Particularly, modeling shapes as level sets of neural networks has recently demonstrated to be an effective geometric surface representation [27],[24],[15],[33]. Furthermore, neural implicit models can be used to represent dynamic radiance fields with applications to high-quality rendering and animation based on real world data [28],[31],[30],[29] .

The study of shape properties through differential geometry leads to a mathematical framework for intrinsic operations on surfaces based on exterior calculus. In this setting it is possible to define operators that change the shape in meaningful ways, for example the mean curvature flow for surface smoothing [4]. However, most of the research in the area of Geometry Processing is in discrete parametric representations, such as polygonal meshes [5],[9],[10].

3 Conceptualization

3.1 Implicit surfaces

The zero-level set $g^{-1}(0)$ of an *implicit function* $g : \mathbb{R}^3 \to \mathbb{R}$ can be used to represent a surface S in $\mathbb{R}^3$. If g is smooth and has $c \in \mathbb{R}$ as a regular value, then its c-level set is a regular surface S. A number $c \in g(\mathbb{R}^3)$ is a *regular value* of g iff $\nabla g \neq 0$ in $g^{-1}(c)$. Conversely, given a regular surface S in $\mathbb{R}^3$, there is a function $g : \mathbb{R}^3 \to \mathbb{R}$ having S as its c-level set, i.e. $g^{-1}(c) = S$ (Prop. 2 in [13, Sec. 2.7]). Therefore, given a sample of points on the surface S, we could try to construct the implicit function g, which this paper considers to be a neural network.

3.2 Neural implicit surfaces

We say that the implicit function g is *neural* iff it is modeled by a neural network. We call the zero-level set $g^{-1}(0)$ a *neural implicit surface*. SIREN [33] and IGR [15] are examples of neural networks capable of representing smooth implicit functions.

Let S be a surface in $\mathbb{R}^3$, and $g : \mathbb{R}^3 \to \mathbb{R}$ be an "unknown" neural implicit function. To compute the parameter set of g such that $g^{-1}(0)$ approximates S, it is common to consider the following *Eikonal* problem in the loss function.

$$\begin{cases} |\nabla g| = 1 & \text{in } \mathbb{R}^3, \\ g = 0 & \text{on } S, \\ \frac{\partial g}{\partial N} = 1 & \text{on } S \end{cases} \tag{1}$$

The *Eikonal equation* $|\nabla g| = 1$ asks for g to be a *signed distance function*. The *Dirichlet condition*, $g = 0$ on S, requires that g is the signed distance from a set in $\mathbb{R}^3$ that includes the surface S. The *Neumann condition*, $\frac{\partial g}{\partial N} = 1$ on S, forces ∇g to be aligned to the normal field N of S. This is due to the fact that $\frac{\partial g}{\partial N} = \langle \nabla g, N \rangle$. Observe that adding such important constraints in the training would require higher-order derivatives of g. Therefore, from now on, we will restrict our study to smooth neural implicit functions.

In this work, we are interested in using neural networks to model animations of the initial implicit surface S. The level sets $g^{-1}(c)$, with $c \in g(\mathbb{R}^3)$, of the neural implicit function g could be used to animate $S \approx g^{-1}(0)$. However, this do not allow intersections between surfaces at different time instants, because the level sets of g are disjoint. To avoid this we extend the domain of the neural implicit function adding parameters to control animations.

3.3 Neural homotopies

$\mathbb{R}^3$ has been considered as the domain of smooth neural implicit functions [33, 15, 25]. However, as mentioned in the introduction (Sec 1), we can augment $\mathbb{R}^3$ by a product space $\mathbb{R}^3 \times \mathbb{R}^k$, where $p \in \mathbb{R}^3$ denotes a point in the space and $c \in \mathbb{R}^k$ denotes a vector that control k direction of possible animations (rigid motion, smoothing, morphing, ...) of the initial implicit surface. In this work we focus on the *space-time* $\mathbb{R}^3 \times \mathbb{R}$. This would allow for implicit surface animations using the control variable time.

Specifically, let $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be a smooth neural function representing an *one-parameter* family of neural implicit functions $f_t : \mathbb{R}^3 \to \mathbb{R}$ defined by $f_t(p) = f(p, t)$. Topologically, the family f_t represents *homotopies* between any two functions $f_{t_0}, f_{t_1} : \mathbb{R}^3 \to \mathbb{R}$, with $t_0, t_1 \in \mathbb{R}$. Thus, we say that f is a *neural homotopy function*. The underlying family of neural implicit surfaces $S_t = f_t^{-1}(0)$ is a *neural animation* of the initial surface $S_0 = S$. Restricted to the interval $[t_0, t_1]$, the surfaces S_t provides a *neural morphing* between the neural implicit surfaces S_{t_0} and S_{t_1}. Observe that these surfaces can contain singularities as their topologies may change over time t. Hart [17] has studied this phenomenon using *Morse Theory*.

3.4 PDEs and homotopies

In the same way that we considered a neural implicit function as a solution of the Eikonal problem (Eq (1)), we could consider the neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be a solution of a general *partial differential equation* (PDE) problem

$$\mathcal{F}\Big(\nabla^n f, \ldots, \nabla^1 f, f, p, t\Big) = 0$$

where $\nabla^k f$ denotes the derivative of order k of f. *Initial-boundary conditions* could also be used. In this work, we consider several definitions of $\mathcal{F}$ to model problems such as animation, smoothing, morphing, keyframe interpolation, etc.

The time t allows a continuous navigation in the neural animation $S_t = f_t^{-1}(0)$. Each instant t also represents a *transformation* $S_0 \to S_t$ of the initial implicit surface S_0.

Again, we could use other domains besides space-time, for example, $\mathbb{R}^3 \times \mathbb{R}^2$. Here $\mathbb{R}^2$ would represent a 2D interface for navigating in the underlying two-parameter family of neural implicit surfaces. However, we will focus on the case of only one extra dimension, i.e., $\mathbb{R}^3 \times \mathbb{R}$, and mostly take the control variable t as time.

4 Neural Homotopy Framework

This section presents the neural homotopy framework that describe neural animations of a given implicit surface based on phenomena represented by PDEs. Specifically, we seek for a neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ satisfying a given PDE problem with its initial condition being a signed distance function from a surface.

We would like for each time t its restriction $f_t : \mathbb{R}^3 \to \mathbb{R}$ to be as close as possible to a signed distance function, that is, an approximation of a solution of the *Eikonal equation* $|\nabla f_t(p)| = 1$. This is not possible in general, since the family of implicit surfaces $f_t^{-1}(0)$ could change the topology, implying singularities ($\nabla f_t = 0$).

In particular, if f_t satisfies the Eikonal equation, $f_t(p)$ is the signed distance of $p \in \mathbb{R}^3$ from the zero-level set $f_t^{-1}(0)$. This property allows us to use the *sphere tracing algorithm* [16, 18] to visualize the level sets of f_t. See [8] for real-time sphere tracing visualizations of neural homotopies.

4.1 Animation

In this section, we present a simple example of animation using PDEs, the *transport equation*, which models the continuous transport of the level sets of an implicit function along a constant direction. Next, we will see that it is a particular case of a more general process, the *level set equation*, which is the classic approach for handling level set animation using PDEs.

4.1.1 Using the transport equation

Let S be a compact surface in $\mathbb{R}^3$. The translation of S towards a vector b can be modeled by a neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ solution of the *transport equation*.

$$\begin{cases} \dfrac{\partial f}{\partial t} + \langle \nabla_p f, b \rangle = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{2}$$

Where g is the signed distance function from the surface S. The spatial gradient of f is denoted by $\nabla_p f := \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$.

Observe that $f(p,t) = g(p - tb)$ is a solution of Equation (2). The function f_t is a signed distance function for each time t and the neural animation $S_t = f_t^{-1}(0)$ represent the movement of S along tb.

For a simple example of how to expand the domain of f, consider it to be $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$. Then update Equation (2) using the transport equation $\frac{\partial f}{\partial t}(p,b,t) + \langle \nabla_p f(p,b,t), b \rangle = 0$. Therefore, given a solution f of the resulting PDE problem, its associated neural animation $S_{(b,t)}$ is embedded in $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$ and encodes all of the rigid motions $p + tb$ of S.

4.1.2 Using vector fields

In this case a family of vector fields drives the animation of the surface S. For this purpose, we use a technique known in the literature as the *level set method*, introduced by Osher and Sethian [26]. This approach is used in several works [37, 11, 21].

In Equation (2) the implicit surface S was transported along the constant vector field b. We can generalize this procedure by replacing the linear movement $p + tb$ of the point p by an *one-parameter* family of deformation $\mathbf{x} : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ satisfying $\mathbf{x}(p,0) = p$. Then, the one-parameter family of surfaces $\mathbf{x}(S,t)$ for $t \in \mathbb{R}$ results in an animation of the initial surface S.

Next, we represent the animation $\mathbf{x}(S,t)$ in terms of neural implicit surfaces. Let g be the signed distance function of S. We seek a neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ such that $f_t^{-1}(0)$ corresponds to $\mathbf{x}(S,t)$ for each $t \in \mathbb{R}$. Therefore, the neural homotopy must satisfies:

- $f = g$ on $\mathbb{R}^3 \times \{t = 0\}$;
- f must be constant along each curve $\alpha(t) = \mathbf{x}(p,t)$ for $p \in \mathbb{R}^3$, i.e. $f(\alpha(t),t) = c$ iff $g(p) = c$.

For each value $c \in g(\mathbb{R}^3)$, the second condition requires that the c-level set of f_t matches with $\mathbf{x}(g^{-1}(c),t)$. To compute f, we observe that for each fixed point p, the function $f(\alpha(t),t)$ is constant. Deriving, we obtain

$$\frac{\partial f}{\partial t}(\alpha(t),t) + \langle \nabla_p f(\alpha(t),t), \alpha'(t) \rangle = 0. \tag{3}$$

This equation is similar to the transport equation (Eq (2)) but with the derivative $\alpha'(t)$ substituting the vector b. The derivative $\alpha'(t)$ corresponds to a vector field along the trajectory $\alpha(t)$ of the point p.

As Equation (3) holds for each point p, we can drop their trajectories α and use only the derivatives α' in the equation. Specifically, we consider these derivatives to be represented by an one-parameter family of vector field $V : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$, with $V(p,t) = \alpha'(t)$ for $p \in \mathbb{R}^3$. Therefore, the unknown neural homotopy f encoding this animation should be the solution the (*neural*) *level set equation*.

$$\begin{cases} \dfrac{\partial f}{\partial t} + \langle \nabla_p f, V \rangle = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{4}$$

Let f be a solution of the neural level set equation. By the above discussion, $f(p,t)$ has the form $g(\mathbf{x}(p,t))$, where $\frac{d}{dt}\mathbf{x}(p,t)$ coincides with $V(p,t)$. In other words, the neural homotopy f implicitly encodes the integration of V. Therefore, defining a family of vector fields is a way to animate a given surface using the level set equation. Section 4.2 present an example where V smooths the surface S.

4.1.3 Generalized level set method

Even though this work is focusing on the space-time case, here we are going to conceptualize a possible generalization of the level set method.

We could augment the domain $\mathbb{R}^3 \times \mathbb{R}$ of the neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ by a product space $\mathbb{R}^3 \times \mathbb{R}^k$, with $k \in \mathbb{N}$, to represent k different animations in a single function. Specifically, let $f : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ be a (unknown) k-parameter family of neural implicit functions. We generalize Equation (4) using

$$\begin{cases} \dfrac{\partial f}{\partial c_i} + \langle \nabla_p f, V_i \rangle = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}^k \\ f = g & \text{on } \mathbb{R}^3 \times \{c = 0\}. \end{cases} \tag{5}$$

Where c_i is the ith coordinate of a vector $c = (c_1, \ldots, c_k) \in \mathbb{R}^k$, and $V_i : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ are k (known) families of vector fields. The k level set equations $\frac{\partial f}{\partial c_i} + \langle \nabla_p f, V_i \rangle = 0$ force f to encode a different animation (indexed by $i = 1, \ldots, k$) on each variable c_i.

4.2 Deformation using the mean curvature flow

This section presents the *mean curvature equation*, a famous PDE studied by the *differential geometry* and the *analysis* communities [4]. The movement of every point, in a family of surfaces that evolves under this equation, has the velocity given by the negative of its mean curvature. Therefore, this family of surfaces is smoothed along time (minimizing distortion). Such properties have significant applications in surface modeling. Here we follow the notations of [1], which presents an implicit formulation for that PDE.

4.2.1 Differential surface properties

To explore the generality of Equation (4) we represent the *mean curvature equation* using it. Consider the family of vector fields $V(p,t) = -H(p,t)N(p,t)$, where $N(p,t) = \frac{\nabla_p f(p,t)}{|\nabla_p f(p,t)|}$ is the normal vector at p of the level set of f_t, and $H(p,t) = \operatorname{div} N(p,t)$ is the *mean curvature*. Replacing V in Equation (4) gives the mean curvature equation.

$$\begin{cases} \dfrac{\partial f}{\partial t} - |\nabla_p f| \operatorname{div}\left(\dfrac{\nabla_p f}{|\nabla_p f|}\right) = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{6}$$

Again, g is the signed distance function from the initial surface S. Intuitively, the resulting family of implicit surfaces $S_t = f_t^{-1}(0)$ is moved in the direction of the mean curvature vector $-H(p,t)N(p,t)$. Therefore, it contracts regions of S with positive curvature and expands regions of negative curvatures. As a consequence, this procedure smooths the initial surface S.

4.2.2 Heat equation

Let f be a solution of Equation (6). If the functions f_t are signed distance functions ($|\nabla_p f| = 1$) the mean curvature equation coincides with the well-known *heat equation* $\frac{\partial f}{\partial t} = \Delta_p f$. Remember the *Laplacian operator* $\Delta_p f = \operatorname{div} \nabla_p f$.

4.2.3 Minimal surfaces

The evolution of a surface under the mean curvature equation produces a family of surfaces that decrease the area. Let $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be an one-parameter family of implicit functions solution of Equation 6, and S_t be its corresponding one-parameter family of surfaces. We can measure the area of these surfaces using $\text{Area}(S_t) = \int_{S_t} dS_t$, where dS_t is the area form of S_t. It can be proved [13, Sec. 3.5], [23, Cor. 6.2] that the *first variation of area* of the family S_t is given by

$$\frac{d}{dt}\text{Area}(S_t)\Big|_{t=0} = -\int_{S_0} H^2 dS_0, \tag{7}$$

Therefore, the area of the family of surfaces S_t is decreasing.

Let S be a surface, and S_t be the corresponding family of surfaces resulting from evolving the mean curvature flow of S. A surface S_{t_0} is *critical* if it satisfies $\frac{d}{dt}\text{Area}(S_{t_0})\Big|_{t=t_0} = 0$, in other words, if its mean curvature is constant equal to zero. This surface is also called *minimal*. Examples of minimal surfaces include the plane, catenoids, helicoids, Enneper surface, Costa's minimal surfaces, etc.

We can add a constraint in the mean curvature equation to fix a given region of the initial surface S along the time. In the case that S has a boundary curve, fixing this curve and evolving Equation (6) leads to a surface of minimal area. This problem is related to the physical shapes of soap films at equilibrium under the surface tension [36].

4.3 Morphing between implicit surfaces

In this section, we present examples of PDEs whose solutions are homotopies (morphing) between implicit functions.

Let S_1 and S_2 be two compact surfaces in $\mathbb{R}^3$, and g_1 and g_2 be their signed distance functions. We start with the case where the morphing of S_1 into S_2 is driven by a vector field.

4.3.1 Morphing driven by a vector field

We define a family of vector fields $V : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ for Equation (4) such that its solution is a morphing between the surfaces S_1 and S_2.

$$V(p,t) = -g_2(p)\frac{\nabla_p f(t,p)}{|\nabla_p f(t,p)|} \tag{8}$$

Therefore, the corresponding PDE problem is obtained by substituting Equation (8) in Equation (4).

$$\begin{cases} \dfrac{\partial f}{\partial t} - |\nabla_p f| g_2 = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g_1 & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{9}$$

Let f be a solution of Equation (9). Intuitively, the homotopy will locally inflate S_1 if inside S_2, and deflate if outside, so that S_1 will always try to fit to S_2 [11]. However, we do not have the exact time t that the homotopy reaches the signed distance g_2 of S_2, i.e. $f_t^{-1}(0) = S_2$. Therefore, using this strategy, we are not able to handle the morphing between more than two implicit surfaces.

4.3.2 Morphing using only the spatial Eikonal equation

We consider the simple PDE problem of asking for the unknown homotopy f between g_1 and g_2 to be a family of signed distance function, i.e, to satisfy the Eikonal equation $|\nabla_p f| = 1$ in the space.

$$\begin{cases} |\nabla_p f| = 1 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g_1 & \text{on } \mathbb{R}^3 \times \{t = 0\}, \\ f = g_2 & \text{on } \mathbb{R}^3 \times \{t = 1\}. \end{cases} \tag{10}$$

Note that the time instants 0 and 1 could be any two numbers. This allows us to generalize Equation (10) to consider the *keyframe* morphing of n surfaces S_i.

$$\begin{cases} |\nabla_p f| = 1 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g_k & \text{on } \mathbb{R}^3 \times \{t = k\}. \end{cases} \tag{11}$$

Observe that Equation (11) admit multiple solutions since no time restriction is considered along the time.

4.3.3 Morphing using the mean curvature equation

We could ask for the morphing of the level sets to minimize area deformation. For this we use the mean curvature equation (Eq (6)) in Equation (11).

$$\begin{cases} \dfrac{\partial f}{\partial t} - |\nabla_p f| \operatorname{div}\left(\dfrac{\nabla_p f}{|\nabla_p f|}\right) = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f = g_k & \text{on } \mathbb{R}^3 \times \{t = k\}. \end{cases} \tag{12}$$

Equation (12) simulates the morphing (minimizing area distortions) between the family of implicit surfaces S_i. Again, g_i are the signed distance functions from S_i.

5 Learning a neural homotopy

Let $\{S_i\}$ be a set of n compact surfaces inside the cube $\mathcal{Q} := [-1,1]^3$ in $\mathbb{R}^3$. Let $f_\theta : \mathcal{Q} \times [-1,1] \to \mathbb{R}$ be a neural homotopy represented by a neural network with an unknown set of parameters θ. This section defines a loss function to estimate the parameters θ such that the resulting homotopy f_θ approximates a solution of a given PDE problem satisfying the conditions $(f_\theta)_{t_i} = g_i(\cdot)$ on $\mathcal{Q} \times \{t = t_i\}$, where $t_i \in [-1,1]$, and g_i are the signed distance functions of S_i at the times t_i.

5.1 PDE problem

We train the parameter set θ by forcing f_θ to be a approximated solution of a given PDE problem.

$$\begin{cases} \mathcal{F}\Big(\nabla^n f_\theta, \dots, \nabla^1 f_\theta, f_\theta, p, t\Big) = 0 & \text{in } \mathcal{Q} \times [-1,1], \\ f_\theta = g_i & \text{on } \mathcal{Q} \times \{t = t_i\}. \end{cases} \tag{13}$$

$\mathcal{F}$ is a smooth function depending on the derivatives of f_θ. The unknown neural homotopy f_θ encodes a certain phenomenon ruled by $\mathcal{F}$. For examples, see Section 4. The interval $[-1,1]$ in $\mathcal{Q} \times [-1,1]$ can be used to control the neural animation $S_t = (f_\theta)_t^{-1}(0)$. The signed distances g_i are the PDE conditions at the times t_i.

5.2 Loss functional

We use Equation (13) to define a loss function to train the parameters of f_θ.

$$\mathcal{L}(\theta) = \underbrace{\int_{\mathcal{Q}\times[-1,1]} |\mathcal{F}| dp dt}_{\mathcal{L}_{\text{PDE}}} + \underbrace{\sum_{i=1}^{n} \int_{\mathcal{Q}\times\{t=t_i\}} |f_\theta - g_i| dp}_{\mathcal{L}_{\text{data}}}. \tag{14}$$

The *PDE constraint* $\mathcal{L}_{\text{PDE}}$ forces the neural homotopy f_θ to satisfy the PDE $\mathcal{F}\big(\nabla^n f_\theta, \dots, \nabla^1 f_\theta, f_\theta, p, t\big) = 0$. This constraint works like a regularization of f_θ requiring it to obey the phenomenon ruled by $\mathcal{F}$. The *data constraint* $\mathcal{L}_{\text{data}}$ asks for f_θ to satisfies the conditions $f_\theta = g_i$ on $\mathcal{Q} \times \{t = t_i\}$ of the PDE problem (Eq (13)).

5.3 Training

To find an approximation f_θ of a solution of Equation (13) we seek a minimum θ of the loss function $\mathcal{L}$ using the *gradient descent*. In practice, we enforce the loss function $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{data}}$ with a sampling in $\mathcal{Q} \times [-1,1]$ and another sampling in $\bigcup_{i=1}^n \mathcal{Q} \times \{t = t_i\}$. In other words, we have to deal with a sampling for the PDE constraint and another one for the data constraints.

5.3.1 Two types of samples

During training, we sample the minibatches of $l_1 \in \mathbb{N}$ *spacetime-points* $\{p_j, t_j\}$, $j = 1, \ldots, l_1$, in the cube $\mathcal{Q} \times [-1, 1]$ randomly. Then, the PDE constraint $\mathcal{L}_{\text{PDE}}$ is enforced in the coordinates (p_j, t_j), yielding a loss function approximation

$$\widetilde{\mathcal{L}_{\text{PDE}}} = \sum_{n=1}^{l_1} \Big| \mathcal{F}\big(\nabla^n f_\theta(p_j, t_j), \ldots, \nabla^1 f_\theta(p_j, t_j), f_\theta(p_j, t_j), p_j, t_j\big) \Big|.$$

Therefore, the *Monte Carlo integration* ensures that the approximation of $\mathcal{L}_{\text{PDE}}$ gets better as the size l_1 of the sampling grows.

The sampling for the data constraint $\mathcal{L}_{\text{data}}$ is slightly different. We follow the definitions of [25], whose sampling strategy is briefly explained here. The data constraint $\mathcal{L}_{\text{data}}$ is responsible of forcing f_θ to fit to the input dataset which, in practice, we consider to be a set of n triangle meshes $\{T_1, \ldots, T_n\}$. Each triangle mesh T_i is a discrete approximation of the surface S_i which we will try to reconstruct as a neural implicit surface $(f_\theta)_{t_i}^{-1}(0)$. In this case, we use the Eikonal equation (Eq (1)) to define $\mathcal{L}_{\text{data}}$. Specifically, we write it as a sum $\mathcal{L}_{\text{data}} = \sum_{i=1}^{n} \mathcal{L}_{S_i}$, with $\mathcal{L}_{S_i}$ managing the approximation of the signed distance g_i of S_i by the neural implicit function $(f_\theta)_{t_i} = f_\theta(\cdot, t_i)$.

$$\mathcal{L}_{S_i} = \underbrace{\int_{\mathcal{Q} \times t_i} \Big| 1 - \big|\nabla (f_\theta)_{t_i}\big| \Big| dp}_{\mathcal{L}_{\text{Eikonal}}} + \underbrace{\int_{S_i} \big| (f_\theta)_{t_i} \big| dS_i}_{\mathcal{L}_{\text{Dirichlet}}} + \underbrace{\int_{S_i} \Big| 1 - \big\langle \nabla (f_\theta)_{t_i}, N_i \big\rangle \Big| dS_i}_{\mathcal{L}_{\text{Neumann}}}.$$

Where $\mathcal{L}_{\text{Eikonal}}$ forces $(f_\theta)_{t_i}$ to be a signed distance function of a set $X \subset \mathcal{Q}$, $\mathcal{L}_{\text{Dirichlet}}$ asks for f_θ to be zero on $S_i \subset X$, and $\mathcal{L}_{\text{Neumann}}$ requires the alignment between the gradient of $(f_\theta)_{t_i}$ and the normal vectors N_i of S_i. To force $X - S_i \approx \emptyset$ we add the signed distance g_i of a sample of points in $\mathcal{Q} - S_i$ in the $\mathcal{L}_{\text{Dirichlet}}$ using $\big|(f_\theta)_{t_i} - g_i\big|$.

In practice, f_θ is supervised using the sample of points $\{p_j, N_j\}_i$, where $\{p_j\}_i$ are the vertices of the triangle meshes T_i, and $\{N_j\}_i$ are their normals. During training, $\mathcal{L}_{\text{Eikonal}}$, $\mathcal{L}_{\text{Dirichlet}}$, and $\mathcal{L}_{\text{Neumann}}$ are discretized into summations using Monte Carlo integration, as in the PDE constraint case. Then, we sample *minibatches* with $l_2 \in \mathbb{N}$ *on-surface* points and $l_3 \in \mathbb{N}$ *off-surface* points. The on-surface points are uniformly sampled in the point cloud $\bigcup_{i=1}^{n} \{p_j, N_j\}_i$, and the off-surface points are uniformly sampled on $\bigcup_{j=1}^{n} \mathcal{Q} \times t_i$.

5.3.2 Samples proportions

During the training of the neural homotopy f_θ, we sample minibatches of $l = l_1 + l_2 + l_3$ points to feed the loss function. Where l_1, l_2, l_3, are the number of spacetime, on-surface, and off-surface points. In our experiments, we obtained important results using $l_1 = \frac{l}{2}$, $l_2 = \frac{l}{4}$, and $l_3 = \frac{l}{4}$.

5.4 Visualization

Let $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be a trained neural homotopy. To visualize the corresponding neural animation $S_t = (f_\theta)_t^{-1}(0)$, we can consider two approaches: *marching cubes* or *sphere tracing*.

5.4.1 Marching cubes

We can use the *marching cubes* [20, 19] to approximate an neural surface S_{t_0} by a triangle mesh T_{t_0} using a sample of $(f_\theta)_{t_0}$ at the vertices of a regular grid in $\mathcal{Q}$. Then the *rasterization* algorithm render views of T_{t_0}. To visualize the neural animation S_t, we repeat this procedure for a sequence of time instants. In Section 6, we use this approach to visualize trained neural homotopies.

The majority of neural implicit works use marching cubes to visualize the resulting neural implicit surfaces. This is probably due to the visualization not being the main focus of those works. However, marching cubes is very sensitive to grid discretization and it is not interactive. The need for high resolution also demands processing, making its performance prohibitive for real-time applications.

5.4.2 Sphere tracing

Assume that f_θ is a family of signed distance functions $(f_\theta)_t$. *Sphere tracing* [16, 18] can be used to visualize interactively the neural animation S_t. It consists of computing the intersections of view rays with S_t. Let p_0 and v be the ray origin and ray direction, respectively. The intersection between the ray $p_0 + sv$, $s \in \mathbb{R}$, and the zero-level set S_t of the signed distance function $(f_\theta)_t$ is approximated by iterating $p_{i+1} = p_i + v f_\theta(p_i, t)$.

[8] present real-time visualizations of neural animations using a multiscale sphere tracing algorithm.

5.5 Neural network architecture

This work considers the network f_θ to be represented by a *multilayer perceptron*

$$f_\theta(p) = W_{d+1} \circ f_d \circ f_{d-1} \circ \cdots \circ f_1(p) + b_{d+1}, \tag{15}$$

consisting of d hidden layers $f_i(p_i) = \varphi(W_i(p_i) + b_i)$, where $W_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ are fully connected layers (linear maps), $b_i \in \mathbb{R}^{N_{i+1}}$ are the biases, and the *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$ is a smooth function. The activation function φ is applied at each coordinate of the vector $W_i(p_i) + b_i$. The parameter θ consists of the union of the coefficients of W_i and the biases b_i. The integer d is the *depth* of f_θ and the dimensions N_i are its layer *widths*.

In our experiments, we use the sine as the activation function, resulting in *sinusoidal networks* [7]. We consider the training framework given in [25] that uses the initialization of the network parameters given in [33].

5.5.1 Gradient computation

The neural network f_θ is smooth since its activation function φ is smooth. We now derive the gradient of f_θ explicitly. This is useful for *shading* computations. For this, we use the chain rule:

$$\begin{aligned}
\nabla f_\theta(p) =& \mathbf{J}\Big(W_{d+1} \circ f_d \circ \cdots \circ f_1(p)\Big) + \mathbf{J}(b_{d+1}) \\
=& \mathbf{J}W_{d+1}\Big(f_d \circ \cdots \circ f_1(p)\Big) \circ \mathbf{J}f_d\Big(f_{d-1} \circ \cdots \circ f_0(p)\Big) \circ \\
& \circ \mathbf{J}f_{d-1}\Big(f_{d-2} \circ \cdots \circ f_1(p)\Big) \circ \cdots \circ \mathbf{J}f_2\Big(f_1(p)\Big) \circ \mathbf{J}f_1(p),
\end{aligned}$$

where $\mathbf{J}$ is the *Jacobian*. Using that $\mathbf{J}b_{d+1} = 0$ and $\mathbf{J}W_{d+1}(q) = W_{d+1}$, we obtain a simpler formula:

$$\nabla f_\theta(p) = W_{d+1} \circ \mathbf{J}f_d(p_d) \circ \mathbf{J}f_{d-1}(p_{d-1}) \circ \cdots \circ \mathbf{J}f_2(p_2) \circ Jf_1(p),$$

where $p_i = f_i \circ \cdots \circ f_1(p)$. Simple calculations lead us to an explicit formula of the Jacobian $\mathbf{J}f_i(p_i) = W_i \odot \varphi'\big[a_i| \cdots |a_i\big]$, where $\odot$ is the *Hadamard* product, and the matrix $\big[a_i| \cdots |a_i\big]$ has N_i copies of the vector $a_i = W_i(p_i) + b_i \in \mathbb{R}^{N_{i+1}}$.

6 Experiments

In this section, we use the above learning framework to approximate a solution of Equations 2, 6, 11, and 12. The details of the experiments presented below can be found in Appendix A.

6.1 Transport of an implicit surface

The first experiment evaluates the learning of the continuous translation of the level sets of an implicit function towards a given direction. For this, we train a neural homotopy to approximate a solution of the *transport equation* (Eq (2)). We know the closed solution of this problem, therefore, we can easily check if the proposed framework can approximate the PDE solution.

Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance function of a surface S inside the cube $\mathcal{Q} = [-1, 1]^3$. The transport of g along a vector $b \in \mathbb{R}^3$ is described by the following PDE (a particular case of Eq. (2)).

$$\begin{cases} \dfrac{\partial f_\theta}{\partial t} + \langle b, \nabla_p f_\theta \rangle = 0 & \text{in } \mathcal{Q} \times [-1, 1], \\ f_\theta = g & \text{on } \mathcal{Q} \times \{t = 0\}. \end{cases} \tag{16}$$

We are assuming the unknown function f_θ in Equation (16) to be a neural homotopy. Therefore, solving this problem is equivalent to finding a parameter set θ such that f_θ satisfies the transport equation. To learn the unknown set of parameters θ, we consider the following loss functional.

$$\mathcal{L}(\theta) = \int_{\mathcal{Q}\times[-1,1]} \left| \frac{\partial f_\theta}{\partial t} + \langle b, \nabla_p f_\theta \rangle \right| dpdt + \int_{\mathcal{Q}\times\{t=0\}} |f_\theta - g| dp \tag{17}$$

The second experiment transports the sphere and the Armadillo model in the direction of $b = (0, 1, 1)$. We are looking for a neural homotopy f_θ satisfying $\frac{\partial f_\theta}{\partial t} + \frac{\partial f_\theta}{\partial x} + \frac{\partial f_\theta}{\partial z} = 0$ with the initial condition $f_\theta(\cdot, 0) = g$, where g is the signed distance of the sphere in the first experiment and the signed distance of the surface of the Armadillo in the second.

6.1.1 Sphere

Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance function of the sphere of radius $\frac{1}{2}$, and $f_\theta : \mathcal{Q}\times[-1, 1] \to \mathbb{R}$ be the neural homotopy trained using the loss function in Equation (17) with the training framework of Section 5.3. Figure 1 presents 4 reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, where $t_i = 0, 0.29, 0.57, 0.86$.

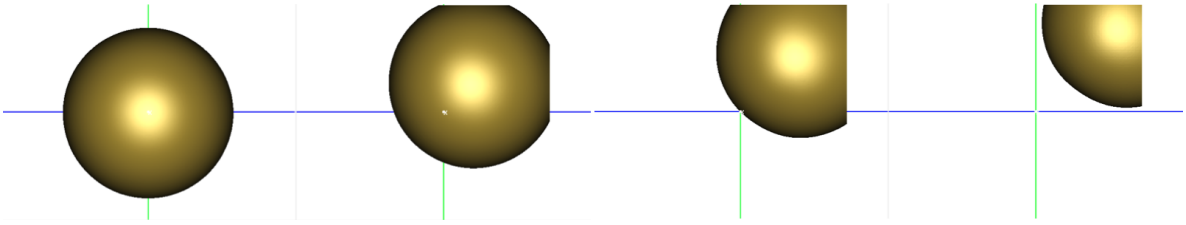

Figure 1: Translation of the sphere along the direction $(0, 1, 1)$.

Observe that the solution of the transport equation is well approximated by the neural homotopy f_θ because the sphere is being translated along the diagonal $(0, 1, 1)$. The spheres are clipped because it leaves the animation domain $\mathcal{Q}$.

6.1.2 Armadillo model

Next, we consider the translation of a surface with more geometrical details. Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance function of Armadillo model, and $f_\theta : \mathcal{Q}\times[-1, 1] \to \mathbb{R}$ be the trained neural homotopy. Figure 2 shows 4 zero-level sets of the neural implicit functions $(f_\theta)_{t_i}$, with $t_i = 0, 0.2, 0.4, 0.6$. As in the sphere case, the solution of the transport equation is well approximated by the neural homotopy f_θ.

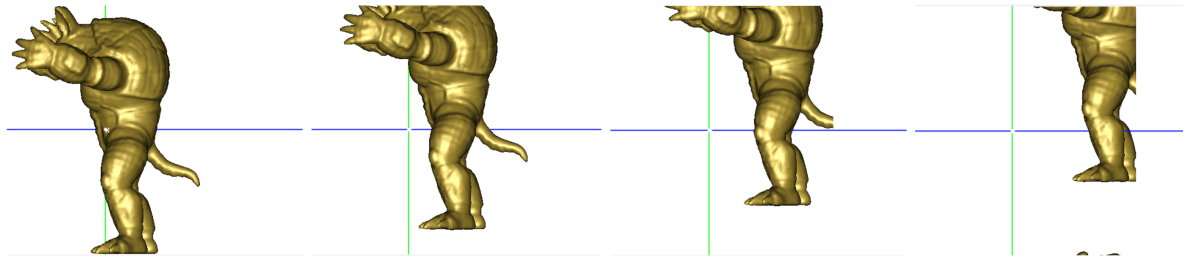

Figure 2: Translation of Armadillo model along the direction $(0, 1, 1)$.

Observe that there is a spurious component below the last Armadillo, however, this is outside the animation domain $\mathcal{Q}$ if we translate it back to the time $t = 0$. This component is probably noise outside of the cube $\mathcal{Q} \times \{t = 0\}$.

6.2 The mean curvature equation

This section presents the experiments for approximating a solution of the mean curvature equation using a neural homotopy.

Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance of a surface S inside the cube $\mathcal{Q} = [-1, 1]^3$. We seek for a neural homotopy $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ that approximates the solution of the mean curvature equation with the initial condition $f_\theta = g$ on $\mathcal{Q} \times \{t = 0\}$. Remember the mean curvature equation $\frac{\partial f_\theta}{\partial t} - |\nabla_p f_\theta| \operatorname{div} \frac{\nabla_p f_\theta}{|\nabla_p f_\theta|} = 0$. Replacing this equation and the signed distance function g in Equation (13) results in a neural version of the mean curvature equation for S.

$$\begin{cases} \dfrac{\partial f_\theta}{\partial t} - \alpha |\nabla_p f_\theta| \operatorname{div} \dfrac{\nabla_p f_\theta}{|\nabla_p f_\theta|} = 0 & \text{in } \mathcal{Q} \times [-1, 1], \\ f_\theta = g & \text{on } \mathcal{Q} \times \{t = 0\}. \end{cases} \tag{18}$$

We added the constant parameter α to control the velocity of the level sets deformation. To learn the unknown neural homotopy f_θ that approximates a solution for this PDE problem, we use the following loss function.

$$\mathcal{L}(\theta) = \int_{\mathcal{Q} \times [-1,1]} \left| \frac{\partial f_\theta}{\partial t} - \alpha |\nabla_p f_\theta| \operatorname{div} \frac{\nabla_p f_\theta}{|\nabla_p f_\theta|} \right| dp dt + \int_{\mathcal{Q} \times \{0\}} |f_\theta - g| dp \tag{19}$$

Next, we present results of the mean curvature flow for the cube, dumbbell, and Armadillo.

6.2.1 Cube

Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance from the cube surface, and $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy. Figure 3 present the zero-level sets of $(f_\theta)_{t_i}$, with $t_i = \frac{i}{5}$, $i = 0, \ldots, 5$. Observe that the solution of the mean curvature equation is well approximated by the trained homotopy f_θ.

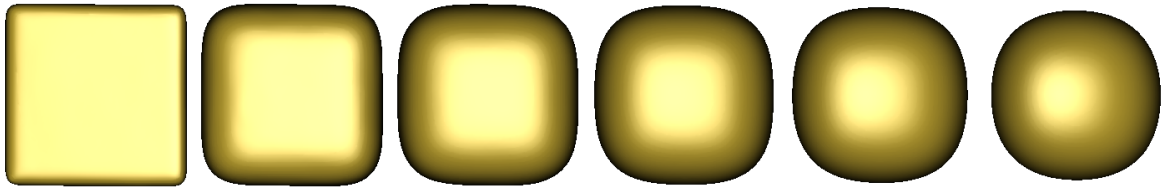

Figure 3: Mean curvature equation of cube surface.

As expected, regions with positive mean curvature, such as the cube corners, have contracted. Remember that the mean curvature equation evolves the level sets toward their normals weighted by the negative of the mean curvature. Therefore, the mean curvature flow of the cube will at some instant of time collapse to a point, but right before it will be very close to a sphere [4].

6.2.2 Dumbbell

Next, we consider a classical example of mean curvature flow. Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance function from a *dumbbell* surface, and $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy. Figure 4 presents zero-level sets of $(f_\theta)_{t_i}$, where $t_i = \frac{i}{10}$ with $i = 0, \ldots, 7$.

Observe that the solution of the mean curvature equation is well approximated by the neural homotopy f_θ. Since the neck region has higher mean curvature, it pinches off first creating two connected components. Later, each component collapses to a point, becoming a small sphere right before. Note that the mean curvature flow of the dumbbell has critical points in different instants of time. These critical points can be studied using Morse Theory [17].

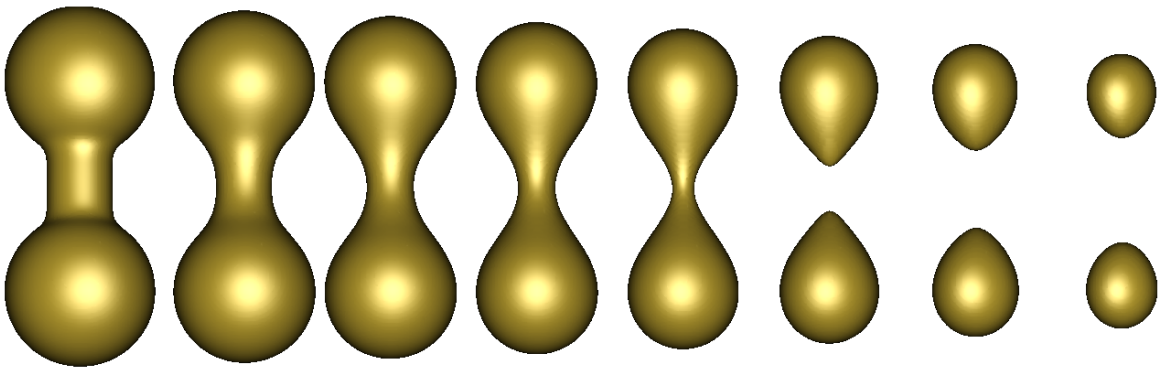

Figure 4: Mean curvature equation of Dumbbell surface.

6.2.3 Armadillo model

Let $g : \mathcal{Q} \to \mathbb{R}$ be the signed distance function from the Armadillo, and $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy. Figure 5 presents reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, where $t_i = -1, -0.5, 0, 0.5, 1$. The Armadillo is at the time $t = -1$, i.e. $(f_\theta)_{-1} = g$ on $\mathcal{Q} \times \{t = -1\}$.

Figure 5: Mean curvature equation of the Armadillo model.

As expected, the regions with positive mean curvature contracted (notice the fingers, for example). This results from the flow smoothing the Armadillo surface.

6.3 Morphing between surfaces

The next experiments consider more than one initial condition, focusing on the case of two conditions. Let $g_i : \mathcal{Q} \to \mathbb{R}$, with $i = 1, 2$, be the signed distance functions from two implicit surfaces S_i. We present two ways to morph g_1 into g_2. First, we consider the Eikonal equation $|\nabla_p f_\theta| = 1$ in $\mathcal{Q} \times [-1, 1]$. That is, we seek for a signed distance function $(f_\theta)_t$ for each time $t \in [-1, 1]$. We use the conditions $f_\theta = g_1$ on $\mathcal{Q} \times \{t_1\}$ and $f_\theta = g_2$ on $\mathcal{Q} \times \{t_2\}$, with $t_1 < t_2$ in $[-1, 1]$.

$$\begin{cases} |\nabla_p f_\theta| = 1 & \text{in } \mathcal{Q} \times [-1, 1], \\ f_\theta = g_1 & \text{on } \mathcal{Q} \times \{t_1\}, \\ f_\theta = g_2 & \text{on } \mathcal{Q} \times \{t_2\}. \end{cases} \tag{20}$$

This problem admit many solutions, since it does not have any time constraint. Then, changing t_1 and t_2 may produce a different homotopy. To optimize the parameters θ we use the following loss function.

$$\mathcal{L}(\theta) = \underbrace{\int_{\mathcal{Q}\times[-1,1]} \big||\nabla_p f_\theta| - 1\big| dp dt}_{\mathcal{L}_{\text{PDE}}} + \underbrace{\int_{\mathcal{Q}\times\{t_1\}} |f_\theta - g_1| dp + \int_{\mathcal{Q}\times\{t_2\}} |f_\theta - g_2| dp}_{\mathcal{L}_{\text{data}}} \tag{21}$$

The second strategy consists of using the mean curvature equation. The resulting morphing should deform S_1 into S_2 minimizing area distortions. For this, we use the mean curvature equation $\frac{\partial f_\theta}{\partial t} - |\nabla_p f_\theta| \operatorname{div} \frac{\nabla_p f_\theta}{|\nabla_p f_\theta|} = 0$ in Equation (13). To learn the homotopy f_θ, we update the PDE constraint using:

$$\mathcal{L}_{\text{PDE}}(\theta) = \int_{\mathcal{Q}\times[-1,1]} \left| \frac{\partial f_\theta}{\partial t} - |\nabla_p f_\theta| \operatorname{div} \frac{\nabla_p f_\theta}{|\nabla_p f_\theta|} \right| dpdt \tag{22}$$

6.3.1 Morph from Cube to Sphere

Let $g_i : \mathcal{Q} \to \mathbb{R}$, $i = 1, 2$ be the signed distance functions of the cube S_1 and the sphere S_2. Let $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy using only the Eikonal constraint (Eq (21)). Figure 6 presents reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, where $t_i = \frac{i}{14}$, $i = 0, \ldots, 7$.

Figure 6: Morphing between the cube and sphere without time constraint.

Surprisingly, even without any time constraint in the PDE, the neural homotopy f_θ morphed the cube ($t = 0$) into the sphere ($t = \frac{1}{2}$). This is probably due to the smoothness (and overfitting capabilities) of the network. It is important to observe that the morphing depends on the chosen time instants $t = 0, \frac{1}{2}$.

Figure 7 illustrates analogous reconstructions considering the mean curvature constraint (Eq (22)) to train f_θ. Observe that using this constraint implies a faster convergence to the sphere. This is a consequence of the fact that the mean curvature flow of the cube already tends to the sphere. We could add a scale α in the mean curvature equation as in Equation (19) to control this deformation.

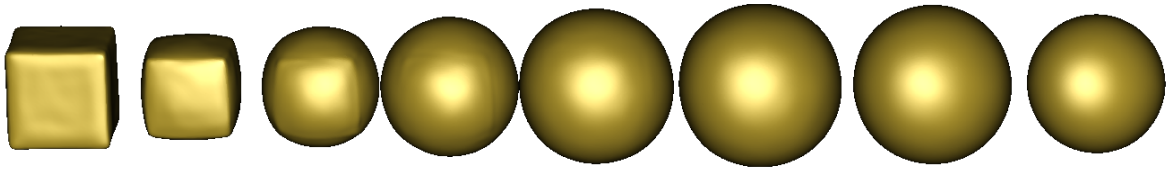

Figure 7: Morph from Cube to Sphere using the mean curvature equation.

6.3.2 Morph from Armadillo to Sphere

Let $g_i : \mathcal{Q} \to \mathbb{R}$, $i = 1, 2$ be the signed distance functions of the Armadillo model S_1 and the sphere S_2, respectively. Let $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy. Figure 8 present the reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, where $t_i = \frac{i^3}{2 \cdot 7^3}$, $i = 0, \ldots, 7$. We use this time sequence to obtain a better illustration of the morphing, since the deformations are stronger near the Armadillo model.

6.3.3 Morph from Woman to Armadillo

In this experiment, we consider two implicit surfaces with similar global geometries. Let $g_i : \mathcal{Q} \to \mathbb{R}$, with $i = 1, 2$, be the signed distance functions of the Woman model S_1 and the Armadillo model S_2, respectively. Let $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy using the Eikonal constraint along time. Figure 9 shows reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, where $t_i = \frac{i}{18}$, $i = 0, \ldots, 9$.

The neural homotopy f_θ successfuly represents the Woman at $t = 0$ and the Armadillo at $t = \frac{1}{2}$. The intermediary zero-level sets of $(f_\theta)_t$, with $t \in (0, \frac{1}{2})$, morph the Woman into the Armadillo. An interesting exercise would be to consider the skeletons of these models and add constraints to keep them fixed over time.

Figure 8: Morphing between the Armadillo and Sphere using the mean curvature equation.

Figure 9: Morphing between Woman and Armadillo models.

6.3.4 Morph from Happy Buddha to Armadillo

To present an experiment using surfaces with different topologies, consider $g_i : \mathcal{Q} \to \mathbb{R}$, $i = 1, 2$, be the signed distance functions of the Happy Buddha S_1 and the Armadillo S_2. Let $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy using the Eikonal constraint. Figure 10 shows reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, with $t_i = \frac{i}{18}$, $i = 0, \ldots, 9$.

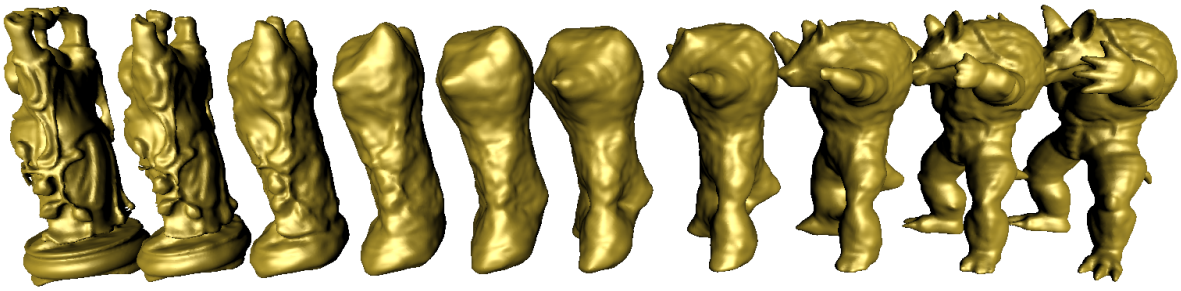

Figure 10: Morphing between Happy Buddha and Armadillo.

Even though these are surfaces with different shapes, f_θ results in a coherent transition between the Happy Buddha and the Armadillo.

Figure 11 illustrates analogous reconstructions using the mean curvature loss function (Eq (22)) to train f_θ. We use the same setting of the previous experiment. Note that this constraint added more surface tension over time.

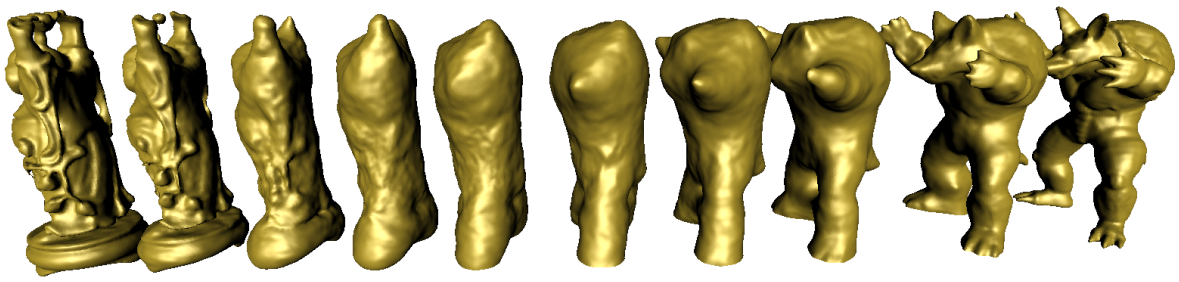

Figure 11: Morphing between Happy Buddha and Armadillo using the mean curvature equation.

6.3.5 Morph from Armadillo to Armadillo

Here, we consider the morphing between two copies of the Armadillo. Let and $f_\theta : \mathcal{Q} \times [-1, 1] \to \mathbb{R}$ be the trained neural homotopy using only the Eikonal constraint. Figure 12 presents reconstructions of the zero-level sets of $(f_\theta)_{t_i}$, with $t_i = \frac{i}{18}$, $i = 0, \ldots, 9$.

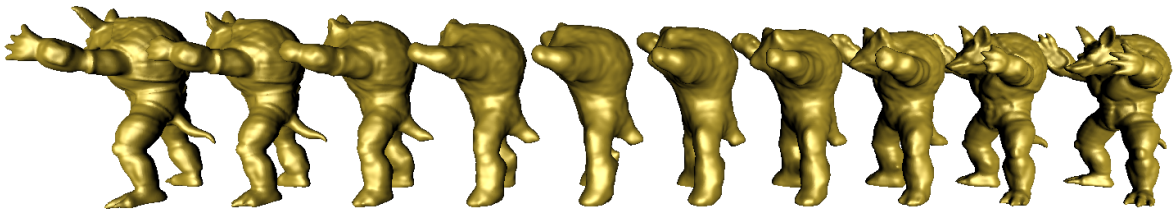

Figure 12: Morphing between two Armadillos using the Eikonal equation.

f_θ is representing the Armadillo, in $t = 0, \frac{1}{2}$, with a good degree of graphic accuracy. However, the intermediary zero-level sets loses details when time moves away from $t = 0, \frac{1}{2}$. Thus, in this setting, f_θ is not capable of representing a constant function over time.

The results are similar when the mean curvature equation is considered. However, we can use $\frac{\partial f_\theta}{\partial t} = 0$ to force f_θ to be constant over time. Figure 13 shows analogous reconstructions using $\frac{\partial f_\theta}{\partial t} = 0$ as constraint to train f_θ. The models are all the same, even spurious artifacts are copied.

Figure 13: Morphing between two Armadillos using the constraint $\frac{\partial f_\theta}{\partial t} = 0$.

The experiments show that the neural homotopy framework can learn a variety of phenomenons, given the right PDE problem to train the neural network.

7 Connecting neural homotopies and neural implicit functions

This section presents a simple way to initialize the parameters of a neural homotopy f_θ based on a trained neural implicit function g_ϕ. It can be used to initialize f_θ in case we would like to approximate it by a solution of a given PDE problem having g_ϕ as initial condition.

Conversely, we give an algorithm to extract a neural function g_ϕ from a neural homotopy f_θ at a given time t.

7.1 Initializing the neural homotopy using a neural implicit function

Let $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ be a neural implicit function with its parameters ϕ trained so that g_ϕ approximates the signed distance function of a surface $S \subset \mathbb{R}^3$. Let $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be an unknown neural homotopy satisfying the PDE problem.

$$\begin{cases} \mathcal{F}\Big(\nabla^n f_\theta, \ldots, \nabla^1 f_\theta, f_\theta, p, t\Big) = 0 & \text{in } \mathbb{R}^3 \times \mathbb{R}, \\ f_\theta = g_\phi & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{23}$$

In Section 6, we have initialized f_θ following the standard definitions of neural networks. However, in this case, we could initialize θ in terms of ϕ such that $f_\theta(p, t) = g_\phi(p)$ for all $t \in \mathbb{R}$. In other words, f_θ will be constant and equal to g_ϕ along the time. To define such initialization, we study the general case: define a neural homotopy $f_\theta : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ such that $f_\theta(p, c) = g_\phi(p)$ for all $c \in \mathbb{R}^k$. For this, we use a neural network f_θ wider than g_ϕ but with the same depth.

Proposition 1. *Let $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ and $f_\theta : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ be neural networks with depth d. If f_θ is wider than g_ϕ, we can define θ in terms of ϕ such that $f_\theta(p, c) = g_\phi(p)$ for all $(p, c) \in \mathbb{R}^3 \times \mathbb{R}^k$.*

Proof. Remember that $g_\phi(p) = B_{d+1} \circ g_d \circ g_{d-1} \circ \cdots \circ g_1(p) + b_{d+1}$, where $g_i(p_i) = \varphi(B_i p_i + b_i)$ is the ith layer, $B_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ is the ith linear map, and $b_i \in \mathbb{R}^{N_{i+1}}$ is the ith bias. Analogously, $f_\phi(p, c) = A_{d+1} \circ f_d \circ f_{d-1} \circ \cdots \circ f_1(p, c) + a_{d+1}$, where $f_i : \mathbb{R}^{M_i} \to \mathbb{R}^{M_{i+1}}$, with $N_i \leq M_i$, given by $f_i(p_i) = \varphi(A_i p_i + a_i)$, is the ith layer of f_θ.

By hypothesis, f_θ and g_ϕ have the same depth d and the width of each layer of g_ϕ is less or equal the width of the respective layer of f_θ. Therefore, define the parameters of the hidden layers using

$$A_i = \left[\begin{array}{c|c} B_i & 0 \\ \hline 0 & I \end{array}\right], a_i = \left[\begin{array}{c} b_i \\ \hline 0 \end{array}\right] \text{ for } 2 \leq i \leq d+1,$$

where $I \in \mathbb{R}^{(M_{i+1}-N_{i+1}) \times (M_i - N_i)}$ is the identity matrix. Therefore,

$$f_i(p, c) = \varphi\left(\left[\begin{array}{c|c} B_i & 0 \\ \hline 0 & I \end{array}\right]\left[\begin{array}{c} p \\ \hline c \end{array}\right] + \left[\begin{array}{c} b_i \\ \hline 0 \end{array}\right]\right) = \left[\begin{array}{c} g_i(p) \\ \hline \varphi(c) \end{array}\right]$$

for $(p, c) \in \mathbb{R}^{N_i} \times \mathbb{R}^{M_i - N_i}$. Note that if $\varphi(c)$ is zero at each layer, we get the desired result. For this, define the first layer of f_θ using

$$A_1 = \left[\begin{array}{c|c} B_1 & 0 \\ \hline 0 & 0 \end{array}\right], a_1 = \left[\begin{array}{c} b_1 \\ \hline 0 \end{array}\right].$$

□

To reproduce Proposition 1 with f_θ deeper that g_ϕ, we would have to deal with hidden layers $f(p_i) = \varphi(Ap + a)$ which does not exist in g_ϕ. Thus, it would be desirable to initialize f as an identity $f(p) = p$. Following the above approach, we would define $A = I$ and $a = 0$ obtaining $f(p) = \varphi(p)$, however, in general, $\varphi(p) \neq p$.

7.2 From a neural homotopy to neural implicit functions

In this subsection, we study the problem of extracting a neural implicit function at a given "time" from a neural homotopy.

Let $f_\theta : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ be a trained neural homotopy represented by the neural network $f_\theta(p, c) = A_{d+1} \circ f_d \circ f_{d-1} \circ \cdots \circ f_1(p, c) + a_{d+1}$, with each ith layer $f_i(p_i) = \varphi(A_i p_i + a_i)$ having a width M_i.

Let $c \in \mathbb{R}^k$, we define a network $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ satisfying $g_\phi(p) = f_\theta(p, c)$ for all $p \in \mathbb{R}^3$. For this, modify the first layer $f_1(p, c) = \varphi(A_1(p, c) + a_1)$ of f_θ. The matrix A_1 has $3 + k$ column vectors $\{w_1, w_2, w_3, u_1, \ldots, u_k\}$ in $\mathbb{R}^{M_2}$, where M_2 is the dimension of the codomain of the first layer f_1. Denoting p by (x, y, z), we obtain

$$A_1(p, c) = x \cdot w_1 + y \cdot w_2 + z \cdot w_3 + \sum_{i=1}^{k} c_i \cdot u_i,$$

where c_i is the ith coordinate of $c \in \mathbb{R}^k$. We use the matrix B_1 consisting of the 3 column vectors w_1, w_2, w_3, and the vector bias $b_1 = a_1 + \sum c_i \cdot u_i$ to construct the first layer g_1 of the neural network g_ϕ. Specifically, we define g_ϕ through the following neural network

$$g_\phi(p) = A_{d+1} \circ f_d \circ f_{d-1} \circ \cdots \circ f_2 \circ g_1(p) + a_{d+1}.$$

Observe that the neural implicit function g_ϕ is the neural homotopy f_θ with its first layer $f_1(p, c)$ replaced by $g_1(p)$, which we define as

$$g_1(p) = \varphi\Big(\underbrace{x \cdot w_1 + y \cdot w_2 + z \cdot w_3}_{B_1 p} + \underbrace{\sum_{i=1}^{k} c_i \cdot u_i + a_1}_{b_1} \Big) = \varphi\big(B_1 p + b_1\big).$$

It follows directly from the definition of g_ϕ that $g_\phi(p) = f_\theta(p, c)$. As a result we have a kind of opposite direction to Proposition 1.

Proposition 2. *Let $f_\theta : \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ be a neural network, and c be a point in $\mathbb{R}^k$. There is a neural network $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ with the same hidden layers of f_θ satisfying $f_\theta(p, c) = g_\phi(p)$ for all $p \in \mathbb{R}^3$.*

8 Conclusions and Future Work

We introduced a framework to explore the differentiable properties of smooth neural implicit surfaces. In this context we extended the representation to higher dimensions which opens up many possibilities to control geometric transformations and shape morphing with applications to animation and modeling.

We gave emphasis to the basic setting of space-time. Future work includes a more thorough investigation of the action of PDEs on surfaces, as well as, generalization of a setting with more dimensions for controlling a wider parameter space of shape variations:

Willmore flow We can try to solve the Willmore flow since there is a level set formulation for this problem [32]. See [6] for important applications in this context.

Generalizing the timespace Replace $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ by $f_\theta : \mathbb{R}^3 \times K \to \mathbb{R}$, where K is a simplicial complex embedded in $\mathbb{R}^m$. Train f_θ such that each vertex of K encodes a known surface.

References

[1] Giovanni Bellettini. *Lecture notes on mean curvature flow: barriers and singular perturbations*, volume 12. Springer, 2014.

[2] D.E. Breen and R.T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[3] M. Cani-Gascuel and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, 1997.

[4] Tobias Holck Colding, William P. Minicozzi II, and Erik Kjær Pedersen. Mean curvature flow. *Bulletin of the American Mathematical Society*, 52(2):297–333, 2015.

[5] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses*, SIGGRAPH '13, New York, NY, USA, 2013. ACM.

[6] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[7] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[8] Vinícius da Silva, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Mip-plicits: Level of detail factorization of neural implicits sphere tracing, 2022.

[9] Fernando De Goes, Andrew Butts, and Mathieu Desbrun. Discrete differential operators on polygonal meshes. *ACM Trans. Graph.*, 39(4), July 2020.

[10] Fernando de Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. Subdivision exterior calculus for geometry processing. 35(4), July 2016.

[11] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for animation. In *Proceedings of the Graphics Interface 1998 Conference, June 18-20, 1998, Vancouver, BC, Canada*, pages 143–150, June 1998.

[12] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 287–290, New York, NY, USA, 1995. Association for Computing Machinery.

[13] Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

[14] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann Publishers, 1998.

[15] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

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

[17] John C Hart. Morse theory for implicit surface modeling. In *Mathematical Visualization*, pages 257–268. Springer, 1998.

[18] John C Hart, Daniel J Sandin, and Louis H Kauffman. Ray tracing deterministic 3-D fractals. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 289–296, 1989.

[19] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15, 2003.

[20] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[21] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995.

[22] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 389–400, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[23] Francisco Martín and Jesús Pérez. An introduction to the mean curvature flow. In *XXIII International Fall Workshop on Geometry and Physics, held in Granada. https://www. ugr. es/jpgarcia/investigacion. html*, 2014.

[24] Mateusz Michalkiewicz, Jhony Kaesemodel Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4742–4751, 2019.

[25] Tiago Novello, Vinícius da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Differential geometry in neural implicits, 2022.

[26] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[28] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *https://arxiv.org/abs/2011.12948*, 2020.

[29] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[30] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Animatable neural radiance fields for human body modeling. *arXiv preprint arXiv:2105.02872*, 2021.

[31] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020.

[32] Martin Rumpf and M Droske. A level set formulation for willmore flow. *Interfaces and free boundaries*, 6(3):361–378, 2004.

[33] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[34] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 335–342, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[35] Thales Vieira, Alex Bordignon, Adelailson Peixoto, Geovan Tavares, Helio Lopes, Luiz Velho, and Thomas Lewiner. Learning good views through intelligent galleries. *Computer Graphics Forum*, 2009.

[36] Stephanie Wang and Albert Chern. Computing minimal surfaces with differential forms. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

[37] Ross T Whitaker. Algorithms for implicit deformable models. In *Proceedings of IEEE International Conference on Computer Vision*, pages 822–827. IEEE, 1995.

A Experiment details

Here, we present the details of the experiments presented in Section 6.

A.1 Transport of an implicit surface

A.1.1 Sphere

This experiment considered the neural homotopy f_θ a neural network with 3 hidden layers $f_i : \mathbb{R}^{128} \to \mathbb{R}^{128}$. We used an input frequency parameter $w_0 = 10$, and trained f_θ with 990 epochs using the loss function in Equation (17).

A point cloud with 32000 points and normals was used, sampled from the sphere of radius $\frac{1}{2}$. During training, We used minibatches with 4000 on-surface points ($g = 0$), 4000 off-surface points ($g \neq 0$), and 8000 points in $\mathcal{Q} \times [-1, 1]$.

A.1.2 Armadillo model

For this experiment, we assumed that the neural homotopy f_θ is represented by a neural network with 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 30$, and trained the neural homotopy f_θ during 1800 epochs using the loss function in Equation (17).

The point cloud had 40000 points, sampled from the surface of the Armadillo model. During training, We used minibatches with 4000 on-surface points ($g = 0$), 4000 off-surface points ($g \neq 0$), and 8000 points in the spacetime domain $\mathcal{Q} \times [-1, 1]$.

A.2 The mean curvature equation

A.2.1 Cube

For this experiment, we assumed that the neural homotopy f_θ is a neural network with 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 16$, and a mean curvature parameter $\alpha = 0.1$. The optimization of θ used 8000 epochs of training using the loss function in Equation (19).

We used a point cloud with 40000 points and normals, sampled from the cube surface. During training, we used minibatches with 5000 on-surface points ($g = 0$), 5000 off-surface points ($g \neq 0$), and 10000 in the spacetime domain $\mathcal{Q} \times [-1, 1]$.

A.2.2 Dumbbell

This experiment considered a neural homotopy f_θ represented by a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 16$. We define the mean curvature parameter $\alpha = 0.05$. We used 2800 epochs of training using the loss function in Equation (19).

We used a point cloud, consisting of points and normals, of size 80000, sampled from the dumbbell surface. During training, We used minibatches consisting of 5000 on-surface points ($g = 0$), 5000 off-surface points ($g \neq 0$), and 10000 in the spacetime domain $\mathcal{Q} \times [-1, 1]$.

A.2.3 Armadillo model

This experiment uses the neural homotopy f_θ given by a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{360} \to \mathbb{R}^{360}$. We used an input frequency parameter $w_0 = 30$. We consider the parameter $\alpha = 0.001$ in Equation (19). We used 33000 epochs of training using the loss function in Equation (19)

We used a point cloud, consisting of points and normals, of size 80000, sampled from the Armadillo surface. During training, We used minibatches consisting of 5000 on-surface points ($g = 0$), 5000 off-surface points ($g \neq 0$), and 10000 in $\mathcal{Q} \times [-1, 1]$.

A.3 Morphing between surfaces

A.3.1 Morph from Cube to Sphere

Here we assumed that the neural homotopy f_θ is a neural network with 3 hidden layers $f_i : \mathbb{R}^{128} \to \mathbb{R}^{128}$. We used an input frequency parameter $w_0 = 20$. We used 650 epochs of training using the Eikonal loss function. For the case using the mean curvature equation we used 6780 epochs of training.

We used two point clouds, consisting of points and normals, each of size 20000, sampled from the (smooth) cube surface and the sphere. During training, We used minibatches consisting of 5000 on-surface points ($f_\theta(\cdot, t) = 0$ with $t = 0, \frac{1}{2}$), 5000 off-surface points ($f_\theta(\cdot, t) \neq 0$ with $t = 0, \frac{1}{2}$), and 5000 in the spacetime domain ($\mathcal{Q} \times [-1, 1]$).

A.3.2 Morph from Armadillo to Sphere

For this experiment We considered the neural homotopy f_θ to be represented by a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 30$. We used 1800 epochs of training using the mean curvature loss function (Eq (22)).

We used two point clouds, consisting of points and normals, each of size 40000, sampled from the Armadillo and the sphere. During training, We used minibatches consisting of 5000 on-surface points ($f_\theta(\cdot, t) = 0$ with $t = 0, \frac{1}{2}$), 5000 off-surface points ($f_\theta(\cdot, t) \neq 0$ with $t = 0, \frac{1}{2}$), and 5000 in the spacetime domain ($\mathcal{Q} \times [-1, 1]$).

A.3.3 Morph from Woman to Armadillo

For this experiment, we considered the neural homotopy f_θ to be represented by a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 30$. We used 9900 epochs of training using the Eikonal loss function (Eq (21)).

We used two point clouds, consisting of points and normals, each of size 40000, sampled from the Woman and Armadillo models. During training, We used minibatches consisting of 3750 on-surface points ($f_\theta(\cdot, t) = 0$ with $t = 0, \frac{1}{2}$), 3750 off-surface points ($f_\theta(\cdot, t) \neq 0$ with $t = 0, \frac{1}{2}$), and 7500 in the spacetime domain ($\mathcal{Q} \times [-1, 1]$).

A.3.4 Morph from Happy Buddha to Armadillo

For this experiment, we considered the neural homotopy f_θ to be a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 30$. We used 1300 epochs of training using the Eikonal loss function (Eq (21)). For the case using the mean curvature equation (Eq (22)), we used 4000 epochs of training.

We used two point clouds, consisting of points and normals, each of size 60000, sampled from the Happy Buddha and Armadillo models. During training, We used minibatches consisting of 4500 on-surface points ($f_\theta(\cdot, t) = 0$ with $t = 0, \frac{1}{2}$), 4500 off-surface points ($f_\theta(\cdot, t) \neq 0$ with $t = 0, \frac{1}{2}$), and 9000 in ($\mathcal{Q} \times [-1, 1]$).

A.3.5 Morph from Armadillo to Armadillo

For this experiment, we considered the neural homotopy f_θ to be represented by a neural network consisting of 3 hidden layers $f_i : \mathbb{R}^{256} \to \mathbb{R}^{256}$. We used an input frequency parameter $w_0 = 30$. We used 1400 epochs of training using the Eikonal loss function (Eq (21)). For the case using the equation $\frac{\partial f_\theta}{\partial t} = 0$, we considered 1400 epochs of training.

We used two point clouds, consisting of points and normals, each of size 40000, sampled from the Armadillo model. During training, We used minibatches consisting of 3750 on-surface points ($f_\theta(\cdot, t) = 0$ with $t = 0, \frac{1}{2}$), 3750 off-surface points ($f_\theta(\cdot, t) \neq 0$ with $t = 0, \frac{1}{2}$), and 7500 in the spacetime domain ($\mathcal{Q} \times [-1, 1]$).