IMPLICIT NEURAL SURFACE DEFORMATION WITH EXPLICIT VELOCITY FIELDS

Lu Sang[1,2], **Zehranaz Canfes**[1], **Dongliang Cao**[3], **Florian Bernard**[3], **Daniel Cremers**[1,2]
[1]Technical University of Munich, [2]Munich Center of Machine Learning
{lu.sang, zehranaz.canfes, cremers}@tum.de
[3]University of Bonn
{dcao, fb}@uni-bonn.de

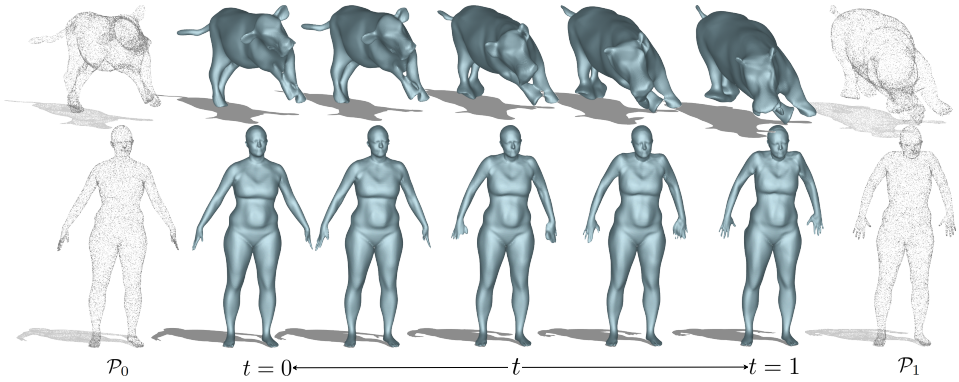

Figure 1: Given two point clouds $\mathcal{P}_0$ and $\mathcal{P}_1$, our method predicts a time-varying neural implicit surface that represents a smooth and physically plausible deformation from $\mathcal{P}_0$ to $\mathcal{P}_1$. To ensure physical plausibility, we utilize a velocity network that leverages smoothness and divergence-free constraints.

ABSTRACT

In this work, we introduce the first unsupervised method that simultaneously predicts time-varying neural implicit surfaces and deformations between pairs of point clouds. We propose to model the point movement using an explicit velocity field and directly deform a time-varying implicit field using the modified level-set equation. This equation utilizes an iso-surface evolution with Eikonal constraints in a compact formulation, ensuring the integrity of the signed distance field. By applying a smooth, volume-preserving constraint to the velocity field, our method successfully recovers **physically plausible** intermediate shapes. Our method is able to handle both rigid and non-rigid deformations **without any intermediate shape supervision**. Our experimental results demonstrate that our method significantly outperforms existing works, delivering superior results in both quality and efficiency[1].

1 INTRODUCTION

Representing surfaces using implicit methods, such as signed distance fields, offers significant advantages over explicit methods in some applications. For example, it allows flexible topological changes and is more memory-efficient compared to storing an explicit representation of a high-resolution surface. Additionally, implicit representations allow for differentiable operations, as the respective surfaces are encoded in smooth fields, which in turn enhances a variety of downstream tasks, such as radiance field rendering by Yariv et al. (2021); Wang et al. (2023). Embedding a signed distance field within a neural network to represent a single surface demonstrated many successful outcomes, such as work from Sitzmann et al. (2020); Gropp et al. (2020); Mescheder et al. (2018). However, using implicit representations to model surface deformation or a dynamic surface evolution, especially with

[1]the code is available: https://github.com/Sangluisme/Implicit-surf-Deformation

physically plausible deformations, still remains challenging. The challenges stem mainly from two inherent characteristics with implicit methods: (i) implicit representations do not store explicit surface point locations, which makes it hard to directly manipulate surfaces during deformation. (ii) the lack of traceable neighboring information in implicit elds prevents the use of ef cient physical constraints, for example, as-rigid-as-possible regularisation, proposed by Sorkine & Alexa (2007), which is crucial in many mesh-based methods such as the work from Alexa et al. (2023); Eisenberger et al. (2021); Cao et al. (2024a). In this paper, we aim to tackle these core problems of implicit surface representations. To this end, we introduce a method that simultaneously recovers implicit neural representations of two given point cloud inputs, together with time-varying intermediate shapes between them. Most notably, our approach distinguishes itself from previous deformation methods based on implicit representations by recovering physically plausible intermediate shapes – without supervision from ground truth intermediate shapes. To achieve this goal, we model the deformation of surface points by training a velocity network that utilizes smoothness and divergence-free constraints, thereby ensuring natural and physically plausible deformations. Our approach circumvents the need for mesh rendering during training, facilitating an end-to-end and fully differentiable training process. Our method supports both intrinsic and extrinsic deformations of the given point clouds, enhancing its versatility and application scope. In summary, we claim the following contributions:

- We propose a novel end-to-end framework that recovers the underlying surfaces of given point clouds together with physically plausible intermediate shapes.
- Our method directly deforms the implicit eld by the explicit velocity eld based on the level-set equation to avoid explicit mesh rendering.
- We propose to use a modi ed level-set equation that combines Eikonal constraint and thereby enables a compact joint optimization while preventing degenerated signed distance elds.
- We validate our method on different datasets and demonstrate that our methods give rise to high-quality interpolations for challenging inputs, both quantitatively and qualitatively.

2 Related Works

Surface representation methods We roughly divide shape representation into explicit and implicit approaches. While explicit representations, such as polygon meshes, store mesh properties, e.g. vertices, edges, and faces explicitly, implicit methods encode the surface information into function elds, such as signed distance elds (SDF). With explicit methods, it is relatively straightforward to edit the shapes, since shape properties can directly be manipulated. However, there are some drawbacks to explicit surface representations. For instance, meshes can only have a xed topology. It is not trivial to adapt vertices and the con guration of their connections (such as edges). Implicit methods, on the contrary, allow arbitrary topological changes since no explicit surface and structural information are stored. Additionally, neural implicit representations enable arbitrary resolutions during inference, without memory increase during storage.

Mesh-based deformation Mesh-based shape deformation is a well-studied problem in computer graphics. The most common strategy is to directly deform vertices based on some local deformation measurements (e.g. as-rigid-as-possible (ARAP) proposed by Sorkine & Alexa (2007), PriMo Botsch et al. (2006), etc.). Another direction is to deform intrinsic quantities like dihedral angles such as work from Alexa et al. (2023); Baek et al. (2015) before reconstructing 3D shapes. Meanwhile, other works formulate shape deformation as a time-dependent velocity eld Charpiat et al. (2007); Eckstein et al. (2007) and incorporate speci c constraints (e.g. volume preservation used by Eisenberger et al. (2018); Eisenberger & Cremers (2020)). Despite the great success achieved by mesh-based shape deformation methods, they rely on the local neighborhood information obtained from edges (or triangles) during deformation. Therefore, shapes have a xed topology during deformation, which limits the applications for shapes with inconsistent topology or partiality. In contrast, our method directly works on an implicit surface, and thus has no constraint on the shape topology and is applicable for partial shapes based on spatial smoothness regularisation. In the experiment part, we demonstrate that our method is capable of deforming shapes with signi cant variations of shape resolution and partiality.

Implicit- eld based deformation Deforming implicit elds presents a signi cant challenge, as all information is encoded within a function eld, preventing direct operations on the object. Previous

works that studied this topic are typically for physical simulation, such as the work of Osher & Paragios (2003); Museth et al. (2002); Jones et al. (2006), and these works use classical discretely stored implicit fields. More recently, the widespread adoption of neural networks to represent implicit fields in shape modeling, work from Yang et al. (2021); Sitzmann et al. (2020); Ma et al. (2020) inspired works to study shape deformation on implicit fields. Several works, e.g., Peng et al. (2021) utilizing neural networks for shape deformation have concentrated exclusively on human body movements , Chen et al. (2021); Deng et al. (2020); Bozic et al. (2021) requiring prior information such as skinning details or intermediate point clouds . Others, like Cao et al. (2024b) have explored deforming implicit fields through generative or diffusion models, but these still necessitate intermediate point clouds for supervision. Some works have trained on datasets of shapes from specific object categories, Deng et al. (2021); Genova et al. (2019); Iglesias et al. (2017); Hao et al. (2020) aiming to deform from one category to another , rather than recovering plausible intermediate shapes. Some works address this problem by defining a latent space and getting a deformed shape via latent space interpolations Liu et al. (2022). Yang et al. (2021) introduced pioneering work that enables direct editing of implicit fields using user-defined handle points, ensuring the deformation remains consistent with the original object. Following this, Mehta et al. (2022) proposed using the level-set equation to deform the implicit network, providing theoretical insights into implicit field deformation. Building on this foundation, Novello et al. (2023) extended these concepts, applying them to 3D shapes, primarily focusing on smoothing surface deformations. In terms of directly deforming the implicit field using a velocity field, their work models the velocity field through linear interpolation between two pre-trained implicit networks representing the target shapes. Therefore, these works are limited to work on a pre-defined velocity field and fail to predict reasonable intermediate shapes when the deformation is not a linear translation.

In this paper, we adopt neural implicit surface representations and tackle the challenging problem of directly deforming the implicit field while recovering physically plausible intermediate shapes – without any rendering or intermediate ground truth supervision. We achieve this by modeling deformations using a velocity field and directly deforming the implicit neural network using modified level-set-equations. Moreover, our training is end-to-end without any pre-trained SDF network needed.

3 METHOD

Given two 3D point clouds $P_0 = \{x_i^0\}_i$ and $P_1 = \{x_i^1\}_i$, we aim to reconstruct a time-varying implicit representation of the inputs together with natural and physically plausible intermediate surfaces. To this end, we adopt a Lagrangian perspective from fluid mechanics to track the trajectory of individual points through a velocity field $V : \Omega \times I \to \mathbb{R}^3$, for $\Omega \subseteq \mathbb{R}^3$ being the point domain, and directly deform the time-varying implicit field $f : \Omega \times I \to \mathbb{R}$, where $I = [0, 1]$ is the time interval. The primary challenges are (i) modeling deformations that are realistic according to physical laws, i.e. modeling physically plausible movements, and (ii) deforming the implicit surfaces without relying on explicit mesh rendering or intermediate shape supervision. To tackle these issues, we employ a smoothness constraint and a divergence-free constraint on the velocity field to ensure realistic movement. Additionally, we utilize a modified level-set equation to directly deform the time-varying implicit surfaces without rendering meshes explicitly.

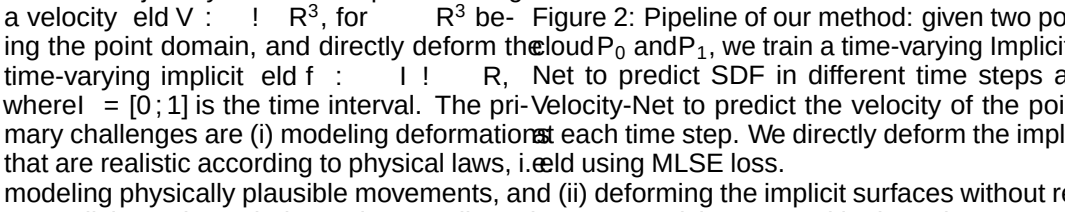

Figure 2: Pipeline of our method: given two point cloud P_0 and P_1, we train a time-varying Implicit-Net to predict SDF in different time steps and Velocity-Net to predict the velocity of the point at each time step. We directly deform the implicit field using MLSE loss.

3.1 TIME-VARYING IMPLICIT FIELDS

The time-varying implicit field $f(\cdot, \cdot)$ takes a point location $x \in \Omega$ and a time $t \in I$ as input, where $I = [0, 1]$ is the time interval and Ω is the surface domain. Its purpose is to encode the evolution of a surface, that is, the shape S_t at time t is represented by the zero-level-set of the implicit function $f(\cdot, t)$, i.e.

$$S_t = \{x \in \Omega \mid f(x, t) = 0\}. \tag{1}$$

Signed-distance fields (SDFs) have many outstanding properties for representing surfaces: for smooth surfaces, the gradient of an SDF coincides with the surface normal direction n on the zero-crossing contour $\partial\Omega$), and the curvature coincides with the divergence, that is C^2

$$n(x;t) = \frac{\nabla f(x;t)}{\|\nabla f(x;t)\|}; \quad \kappa(x;t) = \nabla \cdot n(x;t), \text{ for } x \in \partial\Omega : \tag{2}$$

The time-varying signed distance function f should also satisfy the Eikonal equation, since at every time t, f still is a signed distance field, as proposed by Bothe et al. (2024), i.e. at any time $t \in \mathbb{R}$,

$$\|\nabla f(x;t)\| = 1 : \tag{3}$$

3.2 Velocity Fields

Inspired by the Lagrangian representation in fluid mechanics , which tracks surface points by modeling the particle trajectory $\phi : I \to \Omega$, we track the point by estimating its velocity and integrating the velocity field to form the point trajectory. The trajectory specifically consists of the position of particle x at time t. Assuming the points are moved by an external velocity field $v : \Omega \to \mathbb{R}^3$, where $V \in \mathcal{V}$, and $\mathcal{V}$ is a Hilbert space of a smooth and compactly supported vector field on Ω. The velocity of the moving particle satisfies the following ordinary differential (ODE) equation with the initial condition: the trajectory derivative w.r.t. time is the velocity and the initial point location is given by x^0

$$\begin{cases} V(x) = \frac{d\phi(x;t)}{dt}; & \text{for } t \in I ; \\ \phi(x;0) = x^0 : \end{cases} \tag{4}$$

Note that in our setting, we also enforce the ending point of particle trajectory $\phi(x;1) = x^1$, where x^1 is the point in the target point cloud. To control the smoothness of the movement and ensure physical plausibility, we propose to constrain the velocity field by two aspects: spatial smoothness of the velocity fields, and physical constraints.

Velocity fields that generate diffeomorphisms As the particle path $\phi : t \to \Omega$ represents a trajectory from one point cloud to another, we would like to recover a smooth transformation between two point clouds, which is consistent in both directions. Thus, we seek velocity fields that generate diffeomorphisms when integrated using Eq. (4), i.e. $\phi^{-1}(\cdot;t) = \phi(\cdot;1-t)$

$$\phi(x;t) = x^0 + \int_0^t V(\phi(x;t))dt : \tag{5}$$

Inspired by Dupuis et al. (1998a), this can be achieved by regularizing on the velocity V through the differentiable operator $L = \alpha + \beta I$ such that

$$\|V\|_{\mathcal{V}} = \|LV\|_{l^2} = \int \|LV(x)\|_{l^2} dx ; \tag{6}$$

where I is the identity matrix. A more detailed explanation is provided in the Appendix A.

Divergence-free velocity fields To model physically plausible deformations, we consider the basic conservation laws. One direct conservation law we can borrow is volume conservation. Since we move points on the surface along a trajectory, we assume that no particles are moved across the surface boundary at any time. Then, the total mass inside the surface stays the same, which directly follows from the divergence theorem Kreyszig et al. (2011), i.e.

$$\nabla \cdot V(x) = 0 : \tag{7}$$

A similar idea has also been explored in previous work from Eisenberger et al. (2018); Cosmo et al. (2020) for the case of explicit polygon meshes.

Velocity-Net integration Our smooth velocity field is approximated by an MLP Velocity-Net v_θ. To track the point trajectory, we follow the forward Euler method for integrating ODEs, i.e. we take certain discrete time steps and integrate the velocity step by step using step length $\Delta t = 1/T$. Then, the discrete trajectory of points is formed by

$$\phi(x;t+\Delta t) = \phi(x;t) + V(\phi(x;t))\,\Delta t : \tag{8}$$

The relation between P_0 and P_1 is then given as $\phi(x;0) = x^0,\ x \in P_0$ and $\phi(x;1) = x^1,\ x \in P_1$.

3.3 DIRECT IMPLICIT FIELD DEFORMATION

In the previous sections, we introduced the time-varying implicit fields and velocity fields that represent the shapes and the deformation of points, respectively. In this section, we discuss how to directly deform the implicit field using the external velocity. We borrow the idea from fluid dynamics and treat every point as a fluid particle. Since points stay on the surface of any intermediate shape (i.e. $f(\phi(x;t);t) = 0$ for any $t \in I$ and $x \in \partial\Omega$), it implies there is no in- or out-flow at the surface boundary $\partial\Omega$

$$\frac{d}{dt}\int f(\phi(x;t);t)dx = 0 : \tag{9}$$

Together with the initial condition, that is, the surface deforms from the underlying surface of point cloud P_0, Eq. (9) implies that

$$\begin{aligned} &\partial_t f + V \cdot \nabla f = 0 \quad \text{in} \quad \Omega \times I ; \\ &f(x;0) = f^0 ; \end{aligned} \tag{10}$$

where $f^0(x) = 0$ for $x \in P_0$. The linear transport in Eq. (10) is called the Level-Set Equation (LSE). However, the function f is a signed distance function in our scenario, which means the Eikonal equation in Eq. (3) should also hold to prevent degenerated level-set functions. Previous work from Sussman et al. (1994); Sethian (1996); Sussman & Fatemi (1999) proposed to solve it by introducing a reinitialization equation at a pseudo time τ, e.g. solving

$$\frac{\partial}{\partial \tau} f + \mathrm{sgn}(f^0)(\|\nabla f\| - 1), \quad f|_{\tau=0} = f^0 : \tag{11}$$

However, this requires solving an additional partial differential equation (PDE), and requires that the signed distance field f at time 0 is well initialized. To avoid pre-training a neural network to fit the starting mesh and solve the problem more compactly, we follow the idea of Bothe et al. (2024) and Fricke et al. (2022), combining Eq. (3) and

$$\frac{d}{dt}\|\nabla f(x;t)\| = \|\nabla f\| \left\langle (\nabla V)\frac{\nabla f}{\|\nabla f\|}, \frac{\nabla f}{\|\nabla f\|}\right\rangle \quad 0 ; \tag{12}$$

with Eq. (10) to get our Modified Level-Set Equation (MLSE) that reads

$$\begin{aligned} &\partial_t f + V \cdot \nabla f = \quad {}_{|f|} R(x;t) \quad \text{in} \quad \Omega \times I ; \\ &f(x;0) = f^0 ; \end{aligned} \tag{13}$$

where $R(x;t) = \left\langle (\nabla V)\frac{\nabla f}{\|\nabla f\|}, \frac{\nabla f}{\|\nabla f\|}\right\rangle$. Our MLSE in Eq. (13) preserves the norm of the gradient at the zero-crossing contour. We also adapt the original proposed level-set equation in Bothe et al. (2024); Fricke et al. (2022) by adding a weight $|f|$. We find that it helps to achieve better implicit surfaces while still preserving the desired properties. Compared to the original level-set equation Eq. (10) and discretely enforced Eikonal constraint Eq. (3) on each time step, the modified level-set equation is more compact and solves a single partial differentiable equation (PDE) in an integrated way.

Our MLSE is the bridge between the velocity field and the implicit field. Eq. (13) allows us to deform the implicit field without rendering explicit meshes. Moreover, every component of the formulation is differentiable, thus it enables end-to-end training. Our method jointly recovers the implicit surface for both given point clouds and intermediate shapes without the need for pre-training SDF neural implicit networks for given point clouds or meshes.

3.4 LOSS

We set up the training loss as follows. Velocity-Net loss L_v contains smoothness (Eq. (6)) and divergence-free (Eq. (7)) terms. Implicit-Net loss L_f contains MLSE (Eq. (13) term.

$$\begin{aligned} L_v &= \int \|V\| dx + {}_{\mathrm{div}} \int |\nabla \cdot V| \, dx ; \\ L_f &= \int |\partial_t f + V \cdot \nabla f + {}_{|f|} R| dx : \end{aligned} \tag{14}$$

The divergence-free weight w_{div} can be set to 0 to disable volume preservation. We show examples and the influence of the divergence-free term in the experiment section (Sec. 4).

Finally, to fit the network to the given point clouds, we propose the matching loss

$$L_m = \int_{P_0} |f(x,0)| dx + \int_{P_1} |f(x,1)| dx + \int_{P_0} |\phi(x,1) - x^1| dx : \quad (15)$$

The last term is used to indicate a double integral over x and t, which is needed as $\phi(x,1) = x^0 + \int_0^1 V(\phi(x^0,t),t) dt$. We use the forward Euler step, as described in Sec. 3.2 to integrate during training. Moreover, the last term also implies that one-to-one correspondence is needed for the given two point clouds. However, thanks to the spatial smoothness of the velocity field, we only need a small part of the given correspondence to achieve satisfactory results. Here, P_0 in this P_0 is the set of points for which correspondence information is available. We will discuss the number of correspondences that are needed in Sec. 4. Our total loss term is defined as

$$L = w_f L_f + w_v L_v + w_m L_m ; \quad (16)$$

where w_f, w_v, and w_m are weights to balance the joint training of velocity and implicit function term.

4 Experiments

Neural network architectures and implementation To ensure smooth and diffeomorphic transformations between shapes, as discussed in Sec. 3, we use the following architectures for the two neural networks: (i) Velocity-Net V consists of 8-layer MLP with 256 nodes per layer; (ii) Implicit-Net f also consists of 8-layer MLP with 512 nodes per layer. We use Softplus Zhao et al. (2018) as the activation function. To handle high-frequency information and maintain a diffeomorphism, we incorporate a Lipschitz continuous positional encoding Yang et al. (2021).

Datasets We evaluated our methods using several datasets: Faust Bogo et al. (2014), SMAL Zuffi et al. (2017), SHREC16 Cosmo et al. (2016) and DeformingThings4D Li et al. (2021). Faust and SMAL provide shapes with different categories and movements. Cross-category deformations involve non-rigid transformations between distinct objects, often with significant topological changes. Movement deformation involves changes in gestures within a single object, adhering to physical laws such as rigidity or conformality. DeformingThings4D provides continuous ground-truth displacements of the source mesh vertices in each frame. We used this data set to evaluate our interpolated meshes.

Training strategy To generate training data, we sample 20,000 points on the surface of each mesh to create point clouds with partial correspondences. Each point cloud maintains ground-truth correspondences between 5% to 20% of its points We jointly estimate the velocity and the time-varying signed distance field. To ensure the good initialized deformation of the implicit surfaces, we train 2,000 warm-up epochs only for velocity fields. Then we gradually increase the loss term weight w_f for Implicit-Net. We implement our code using Jax Bradbury et al. (2018) to enable fast higher-order derivative computations. We train for a total of 10,000 epochs with batch size 4,000. The run time is approximately 20 minutes on a GeForce GTX TITAN X GPU with CUDA for each pair.

4.1 Shape Deformation

We test our method on various deformation scenarios and compare it with other methods: LipMLP from Liu et al. (2022) uses MLP layers that satisfy Lipschitz continuity to ensure smooth transitions between source and target shapes. NFGP from Yang et al. (2021) deforms shapes based on a source mesh and user-defined handle points. They estimate the implicit neural surface of the source mesh and then compute the deformed surface to match the target handle points. That means the method requires separate training for each time step. NISE fits two neural networks to the source and target meshes and trains an implicit field with a time dimension to estimate intermediate deformations. We use ground truth meshes to train both NISE and NFGP. NISE takes about 1.5 hours, and NFGP takes around 15 hours for each deformation step (over 75 hours for five steps). Our method requires only 1/5 of the time compared to NISE, excluding the pre-training time for SDF networks of the source and target meshes, which takes around another 20 minutes.

Extrinsic (pose) deformation Extrinsic deformation refers to only pose changes. The transformations occurring within the same object category and no changes to the object type. In this context, we incorporate a divergence-free constraint (c.f. Eq. (7) with $_{div} > 0$) to ensure that only the object volume does not change during the deformation. Fig. 3 illustrates the results on the Faust dataset, benchmarked against other methods. While other methods fail to produce physically plausible intermediate shapes, both NFGP Yang et al. (2021) and our method successfully recover reasonable shapes. However, NFGP requires user-de ned handle points for each step and must be trained incrementally, preventing it from generating a smooth deformed implicit neural surface.

Instrinsic (non-rigid) deformation Non-rigid deformation typically involves different objects for the source and target point clouds. In this case, we disable the divergence-free constraint by setting $_{div} = 0$. Fig. 4 shows an example that includes different categories and poses deformation (intrinsic and extrinsic) in the source and target point clouds. Since NFGP Yang et al. (2021) cannot handle non-rigid deformation, we only provide qualitative visualization results compared with the other two methods. While all methods can recover the source and target meshes, the comparison methods fail to generate reasonable intermediate meshes.

Quantitative evaluation To quantitatively evaluate the interpolated meshes, we use the fox and bear animation from the DeformingThings4D Li et al. (2021) dataset. These examples contain relatively small deformations per frame, making them suitable as ground truth baseline. Each sequence contains 55 meshes. For each dataset, we select 5 key meshes and estimate the deformations between them. We calculate the Chamfer Distance (CD) and Hausdorff Distance (HD) of the recovered meshes compared to the ground truth. We compare our method with LipMLP Liu et al. (2022) and NISE Novello et al. (2023). Fig. 5 shows the average error table over the 55 interpolated meshes (left) and the error plot for each mesh in the bear dataset (right). While all methods accurately recover the mesh at input time steps, LipMLP and NISE exhibit increasing errors at intermediate steps. Our method maintains a consistently low error on the intermediate meshes. Visualization results and error plots for the fox dataset are provided in Appendix A.

4.2 Incomplete and Sparse Input

Implicit methods showcase remarkable exibility in representing shapes with varying typologies. In this section, we show some challenging cases that our method can still tackle.

Different sparsity inputs Most existing approaches require tting two separate networks to estimate the Signed Distance Fields (SDF) for the initial and nal shapes, as highlighted in previous works Novello et al. (2023); Yang et al. (2021); Liu et al. (2022). Consequently, the quality of the results is heavily dependent on the characteristics—such as the sparsity of the source and target point clouds. If the tting process for one shape is unsuccessful, these methods fail to estimate intermediate shapes. Our approach overcomes these challenges through the Velocity-Net, which enables tracking the dense initial point cloud P_0 to the sparse nal point cloud P_1, and recovering the underlying shapes without compromising result quality. Fig. 7 illustrates the varying sparsity levels of the input data. The source point cloud P_0 contains 20;000 points while the target point cloud P_1 only has 2;000 points. While the source point cloud is dense enough to train a neural network for tting an SDF, the target point cloud is much sparser than the source. Although one could employ densi cation

Figure 3: Experiment on extrinsic deformation. LipMLP Liu et al. (2022) and NISE Novello et al. (2023) fail to estimate the physically plausible intermediate shapes. NFGP Yang et al. (2021) can recover reasonable meshes but it is trained separately for each time step. Our method can recover realistic intermediate shapes in one model.