Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Robust Curve Modeling Approach on Surface Meshes via Physics-Informed Neural Networks

ARTICLE INFO

Article history:

Surface meshes, Implicit curves, Physics-informed neural networks, Robust modeling

ABSTRACT

Traditional implicit curve modeling methods on surface meshes, such as variational approaches, are often plagued by numerical instability and heavy reliance on mesh quality, severely limiting their reliability in practical applications. To address these challenges, we propose Neural Implicit Curve Modeling on Meshes (NICMM), a novel framework that integrates Physics-Informed Neural Networks (PINNs) with geometric constraints for robust curve design. NICMM leverages physics-driven loss functions to encode positional, smoothness, and feature-aware constraints, effectively mitigating numerical divergence caused by low-quality meshes. The framework introduces a two-stage training strategy combining pre-training with rapid convergence optimization, and incorporates specialized modules (e.g., Efficient Channel Attention and Light GLU) to enhance feature extraction and computational efficiency. Extensive experiments on the SHREC16 dataset demonstrate that NICMM outperforms traditional variational methods in robustness, achieving high-fidelity curves even on degraded meshes with elongated or near-degenerate elements. Furthermore, NICMM supports feature-aware curve design, enabling alignment with user-specified regions and obstacle avoidance through a unified guidance mechanism. This work establishes a new paradigm for manifold curve modeling, with significant potential in CAD/CAM systems, virtual surgery, and other domains requiring precise and adaptive geometric design.

1. Introduction

Curve design is a classical and fundamental topic in Computer-Aided Geometric Design (CAGD) and Computer Graphics (CG), boasting extensive applications across diverse domains. It plays a crucial role in industrial product design and manufacturing, powering CAD/CAM systems, and is also fundamental in graphics and image processing. After decades of dedicated research, the theories and methodologies for curve design within Euclidean space have matured significantly, reaching a state of well-established sophistication. Nevertheless, this field continues to evolve dynamically, attracting sustained research attention in recent years. Notably, there has been a surge of interest in curve shape control methods, as evidenced by studies such as [35, 39, 24], and their practical applications, as explored in [25].

Driven by pressing industrial demands and rapid advancements in hardware and software technologies—especially the burgeoning field of Artificial Intelligence Generated Content (AIGC), the acquisition and utilization of 3D models have become increasingly accessible and widespread. This digital revolution has, in turn, spurred the in-depth development of theories and techniques in digital geometry processing centered around 3D models. Among the plethora of emerging research topics, curve design on non-Euclidean surface meshes, which involves generating curves embedded within a given curved space, has emerged as a particularly important and vibrant area of inquiry [28, 13, 18]. This research endeavor holds profound theoretical significance and a broad spectrum of practical applications, including mesh segmentation and cutting [33, 5], Voronoi diagrams computation on manifold [36], shape analysis, virtual surgery [31], numerical control (NC) tool path generation [15], and vector graphs [22].

Despite its potential, designing curves on discrete curved 2-manifolds, such as surface meshes, presents a unique set of challenges. Unlike free-form curves in Euclidean space, these curves are typically represented as piecewise-linear polylines intricately embedded within the manifold structure. Existing approaches to this problem can be systematically categorized into two primary paradigms: explicit and implicit methods. Explicit methods directly model the curve using techniques such as projection [11], smoothing [16], parameterization [17], or spline-based approaches [20]. These methods offer a degree of flexibility comparable to that of a Euclidean curve design while striving to satisfy constraints related to smoothness, manifold embedding, and interpolation. However, in their attempts to enforce the critical manifold constraints, they often encounter issues such as compromised curve quality, manifested as local distortions, poor robustness due to high sensitivity to mesh noise, and limited numerical stability, which is heavily dependent on the quality of the input mesh. In contrast, implicit methods operate by constructing scalar fields through the solution of partial differential equations (PDEs) or variational formulations [1] and subsequently extracting level-set curves from these fields. By their very nature, these methods bypass manifold constraints and the problem of self-intersections, enabling the generation of high-quality curves [38]. Nevertheless, since they rely on non-linear numerical computations performed on discrete meshes, their numerical stability is highly contingent upon mesh quality. Poorly shaped mesh elements can easily lead to numerical instabilities, thereby undermining the overall robustness of the algorithms.

To overcome these persistent challenges, our research diverges from traditional numerical optimization-based strategies and delves into the realm of physics-driven geometric deep learning and modeling for implicit curve design on surface meshes. Central to our approach is the utilization of implicit modeling, which effectively sidesteps the complexities associated with manifold constraints and self-intersections, issues that frequently degrade curve quality and inflate computational complexity. Moreover, by harnessing the Physics-Informed Neural Network (PINN) framework [30] for geometric deep learning, we introduce a novel network architecture named Neural Implicit Curve Modeling on Meshes (NICMM). This network not only enables fine-grained multidimensional control over implicit curves but also significantly enhances the overall robustness of the algorithm, marking a significant advancement in the field of curve design on surface meshes.

2. Related Work

Designing curves on manifold surfaces constitutes a fundamental and longstanding problem within the realm of geometric computation. In light of the diverse ways manifold surfaces can be represented, existing research on this topic can be comprehensively classified into two principal approaches. The first approach pertains to curve design on smooth manifolds, typically modeled using parametric surfaces, while the second focus on discrete manifolds, which are commonly represented as meshed surfaces. For the purposes of this paper, our exclusive focus lies on the latter, namely, curve design on surface meshes. This area of research can be further delineated into explicit and implicit methodologies, each with its own unique characteristics and applications.

2.1. Explicit Modeling Methods

Explicit methods directly represent and model curves, capitalizing on the rich corpus of concepts and techniques established for curve design within Euclidean spaces. These methodologies have emerged as the dominant paradigm for curve modeling in discrete manifolds. The core challenge lies in ensuring that the designed curves not only uphold fundamental geometric properties but also adhere rigorously to the constraints imposed by the manifold surface. Typically, these approaches employ intrinsic or extrinsic optimization strategies to sculpt the curves, incorporating a diverse array of techniques such as parameterization, smoothing, projection, and spline extension.

Parameterization-based approaches map the discrete manifold surface onto the Euclidean plane, design the curve within the Euclidean domain, and then map it back to the original surface. For example, Lee et al. [17] performed local parameterization of the regions surrounding the initial curve and utilized the ”geometric snake” model to evolve the curve shapes in the parameter domain. Building on this concept, Lee et al. [18] proposed using an energy formulation based on ”intelligent scissors” to design curves in the local Euclidean space, which were subsequently applied for mesh cutting operations. Although these methods are generally efficient, they are mainly limited to the design of curves in local areas. Moreover, they are prone to parameterization-induced distortion, which can significantly degrade the quality of the curve when applied to larger areas.

Smoothing-based methods define energy functionals that directly characterize the shape of curves on surface meshes and evolve these curves through optimization processes. Jung et al. [12] extended the active contour model originally developed for images to surface meshes, proposing an energy function that integrates both the curve geometry and the mesh features for curve optimization. However, the generated curves are restricted to mesh edges, resulting in suboptimal smoothness. Ji et al. [10] improved the active contour model by relaxing the constraint of curve nodes on mesh edges and introduced topological operations such as splitting, moving and removal of nodes during geometric optimization to enhance curve smoothness. Lawonn et al. [16] constructed a Laplace operator for mesh curves, aiming to reduce geodesic curvature via Laplacian smoothing; however, this approach requires extensive iterations, incurs high computational costs, and struggles to support interpolation constraints. More recently, Pawellek et al. [28] introduced a distance-based smoothing method, which embeds triangular meshes in four-dimensional Euclidean space and computes geodesics on the lifted surface to achieve curve smoothing, which ensures convergence and maintains a close approximation to the initial curves. Despite their general robustness, these techniques face challenges in precisely defining energy functionals for effective curve control due to manifold constraints. Additionally, their high computational complexity often leads to vulnerability to suboptimal local minima, degrading the quality of the curves.

Spline extension methods seek to generalize spline theories from Euclidean space to discrete manifolds under Riemannian metrics. Mancinelli et al. [24] recently provided a comprehensive review of such approaches. Wallner et al. [31] initially replaced Euclidean linear averages with geodesic means using geodesic metrics, generalizing subdivision curve methods to manifold surfaces, albeit with relatively time-consuming computations. Morera et al. [26] developed subdivision curves based on a novel geodesic averaging method that reduced the number of geodesic computations. Mancinelli et al. [23] designed Bézier curves in discrete manifolds by employing an improved recursive De Casteljau algorithm and the Lane - Riesenfeld subdivision scheme, effectively addressing discontinuity issues and enhancing computational efficiency. Their group also used Riemannian straight-edge and compass constructions to generate vectors on surfaces [21]. More recently, they [23] proposed a Newton-based method for computing Riemannian centroids on mesh surfaces, which was then used to construct rational Bézier and B - spline curves on surfaces. Overall, these methods offer limited shape control capabilities and are generally ill-suited for interpolatory curve design.

Extrinsic projection methods tackle the curve design problem in Euclidean space and subsequently project the results onto the manifold, enhancing the flexibility and controllability of curve design. Hofer et al. [8] proposed a discrete spline curve interpolation method based on energy minimization. This approach discretizes curves into polygons, constructs spline energy, and employs a projected gradient method to perform iterative optimization within the surface's ambient space. Pottmann et al. [29] expanded this concept by introducing discrete higher-order spline energies to fit curves on surfaces. Panozzo et al. [27] embedded the manifold surface into a high-dimensional Euclidean space and computed the Riemannian centroid of the corresponding manifold points using the Euclidean centroid and the Phong projection. Although this method can generate smooth curves without iteration, the high-dimensional embedding process is computationally intensive, and projection operations may introduce curve discontinuities. Jin et al. [11] proposed a curve design method based on thin-shell space, enhancing robustness and efficiency, although the construction of the shell space remains complex and time-consuming. More recently, Xu et al. [34] developed a B-spline curve design method using a simplified shell space, improving the quality of curves at the cost of reduced efficiency. By relaxing manifold constraints, these methods increase degrees of freedom in curve design for more flexible control. However, projection operations often compromise algorithmic robustness and efficiency, potentially degrading the quality of the generated curves.

2.2. *Implicit Modeling Methods*

Implicit methods, also known as "level set methods" or "implicit function methods", constitute a modeling paradigm that differs substantially from its explicit counterparts. Instead of direct curve construction, these methodologies focus on generating specific scalar fields over discrete manifolds and then extracting level sets of a predetermined value to derive the target curves. Level-set methods have achieved remarkable feats across diverse disciplines, especially in image and geometry processing. Their extensive applications span image segmentation, surface modeling and reconstruction, mesh smoothing, topology optimization, and path planning, as comprehensively reviewed in [6]. Generally, implicit methods are predominantly based on numerical techniques; however, with the rapid rise of deep learning, learning-based approaches have emerged as a burgeoning research frontier, demonstrating unique advantages. However, their practical implementations remain largely restricted to applications in images such as segmentation, as explored in [9, 32], and are often hampered by limited user control.

The implicit curve methods are evolving with notable advances. Wu et al. [1] pioneered an implicit curve evolution method grounded in the geodesic curvature flow equation. To boost convergence, they adopted a semiimplicit integration technique; however, the method's overall computational efficiency remained sub-par. Building on this foundation, Zhang et al. [37] symmetrized the coefficient matrix of the equation and reduced the computational dimension by establishing a narrow-band domain, thereby significantly enhancing the efficiency of the method and making it more amenable to mesh segmentation tasks. Liu et al. [19] further innovated with an advanced discrete approach for geodesic curvature flow, which made the coefficient matrix sparser and minimized the dimension of the solution through a narrow-band construction, leading to a marked improvement in efficiency. Most recently, Zhang et al. [38] developed a variational implicit curve design method that bypasses a direct solution of the curvature flow equation. By leveraging a variational framework, this method enables the seamless integration of diverse geometric constraints, yielding robust, high-quality curves. Nevertheless, its reliance on numerical techniques renders the approach sensitive to mesh quality.

Implicit methods inherently excel at satisfying manifold constraints, consistently producing high-quality self-intersection-free curves. However, the majority of existing implicit approaches, which depend on diffusion flows and numerical optimization, are inevitably constrained by the limitations of numerical computation. Consequently, their robustness is highly contingent on the quality of the input mesh, posing a persistent bottleneck for practical applications.

2.3. *Physics-Informed Neural Networks*

In recent years, propelled by the exponential growth of deep learning, Physics-Informed Neural Networks (PINNs) have emerged as an innovative paradigm for solving partial differential equations (PDEs), rapidly gaining ground in research interest [14]. By seamlessly integrating the prowess of deep learning with fundamental physical laws, PINNs have demonstrated remarkable problem-solving capabilities. These networks construct residuals derived from the governing equations and boundary conditions of PDEs, which are then incorporated into the loss function. Through the iterative minimization of this loss function, the neural network parameters are optimized, effectively enabling the solution of complex PDE systems. This approach not only significantly reduces the reliance on large volumes of labeled data but also enhances the model's generalization ability, thereby increasing its practical utility across diverse applications. Moreover, PINNs leverage automatic differentiation techniques, facilitating highly accurate derivative computations that are essential for tackling intricate PDEs.

The applications of PINNs have proliferated in a wide range of disciplines. In fields such as fluid mechanics and computational mechanics, PINNs have proven highly effective in solving various PDEs, including the Burgers equation and the Navier-Stokes equations. They exhibit particular strengths in tackling high-dimensional and nonlinear PDEs, where traditional numerical methods often encounter difficulties. Moreover, PINNs can be synergistically integrated with advanced techniques such as proper orthogonal decomposition and discrete empirical interpolation methods to further enhance the accuracy and efficiency of the solution process. For instance, Sahli Costabal et al. proposed Δ-PINNs, which leverage Laplace-Beltrami eigenfunctions as a form of positional encoding. This enables neural networks to better capture the topological structure of complex domains, successfully solving Eikonal and heat equations in challenging geometries such as coils, heat sinks and the Stanford bunny[2]. Additionally, Gao et al. introduced PhyGeoNet, a physics-informed geometry-adaptive convolutional neural network that efficiently solves parametric steady-state PDEs, such as the heat equation and steady-state Navier-Stokes equations, in irregular domains[4].

Inspired by these successful applications and the core principles of PINNs, we explore a PINN-based framework for implicit curve design on surface meshes. The overarching goal is to address the issue of numerical robustness found in existing methods and to advance the state-of-the-art in this critical area of geometric computation. Recent developments have demonstrated the potential of PINNs for handling complex geometries.

3. Proposed Method

Given a discrete triangular mesh $\mathcal{M} =< V, F >$ (where $V$and F denote the vertex set and the face set, respectively) and a user-specified set of control points $\mathcal{P} = \{\mathbf{p}_i\} \in \mathcal{M}$, the goal is to generate a visually smooth curve on the surface that interpolates these control points. Since this work adopts an implicit modeling approach, the task is to solve a scalar field on the mesh and extract its zero level set as the target curve C. Therefore, the resulting implicit curve C must satisfy the following constraints:

1. Positional constraint: $\forall \mathbf{p}_i \in \mathcal{P}, \phi(\mathbf{p}_i) = 0$, where ϕ is the predicted level set function and $\mathbf{p}_i$ are the control points.
2. Geometric smoothness: The curve $C = \phi^{-1}(0)$ should exhibit a continuous normal variation.

To address this challenge, Zhang et al. [38] proposed an efficient variational method. However, this approach exhibits high sensitivity to mesh quality, as its solution hinges on the numerical stability of differential operators (e.g. the Laplace-Beltrami operator). When the mesh contains low-quality elements (e.g., highly skewed triangles), the condition number of the system matrix increases drastically, leading to failure in satisfying the prescribed constraints. Unlike variational approaches, we integrate implicit methods with deep learning and propose a curve design network based on the PINN (Physics-Informed Neural Network), named NICMM (Neural Implicit Curve Modeling on Meshes). By encoding both positional constraints and geometric smoothness constraints into the loss function, NICMM enables end-to-end optimization, thereby circumventing the error accumulation inherent in traditional multistage solvers and significantly enhancing robustness against low-quality meshes. The key components of the proposed approach are elaborated in the following sections, including: (1) construction of input features; (2) network architecture design; and (3) formulation of loss functions.

3.1. Input Feature Construction

Classic Physics-Informed Neural Network (PINN) methods traditionally rely on coordinate points as the sole input to the network. To enhance feature extraction capabilities and accelerate network prediction, we expand the input feature set beyond coordinates, explicitly incorporating constraint encodings and geometric mesh features.

The process begins with the construction of a sparse interpolation scalar field ϕ_p, which is based on the control point sequence $\mathcal{P} = \{\mathbf{p}_i\}_{j=1}^{k}$. Control points within the sequence $\mathcal{P}$ that are not part of the original set of vertex V are treated as virtual vertices and inserted into the mesh $\mathcal{M}$, resulting in an updated mesh structure. In detail, if $\mathbf{p_i}$ is located inside a triangle face, the triangle is virtually split, giving rise to three new triangles. In contrast, when $\mathbf{p_i}$ lies on an edge, we virtually split the edge. Through this systematic procedure, the control points are seamlessly integrated into the mesh as new vertices. A visual representation of control points $\mathbf{p_i}$, highlighted in red, at various positions within the mesh is depicted in Fig. 1.

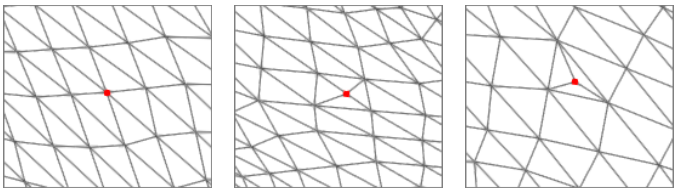
(a) Control point on vertex (b) Control point on edge (c) Control point on face

Fig. 1: Illustration of control points in different Positions

After performing the aforementioned operations, the original mesh $\mathcal{M} = \langle V, F \rangle$ is transformed into an updated mesh $\mathcal{M}' = \langle V', F' \rangle$, where all control points are incorporated into the vertex set V' of $\mathcal{M}'$, ensuring that the control points are precisely positioned as vertices of the mesh.

Subsequently, a discrete interpolation scalar field ϕ_p is constructed. Mathematically, this scalar field is defined as:

$$\phi_p(v) = \begin{cases} 1, & \text{if } v \in \mathcal{P} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

This scalar field acts as an explicit marker for user-defined constraint locations. By clearly denoting these positions, we establish well-defined boundary conditions that are essential for the subsequent physics-guided optimization.

To further enhance efficiency and accelerate convergence, an initial Signed Distance Field (SDF) is precomputed and employed as the foundation for network initialization. The construction of this initial SDF involves the following steps: First, a closed-loop path (represented as a polygonal curve) that traverses the control points is generated by leveraging the mesh edges associated with $\mathcal{P}$. This closed-loop path serves as a constraint for constructing the initial SDF, where the points inside the loop are assigned negative values and the points outside are assigned positive values. Algorithms such as Breadth-First Search (BFS) or the Dijkstra shortest-path algorithm can effectively generate the closed-loop path. The SDF itself can be efficiently computed using the heat method [3], treating all vertices on the closed loop as source points to generate the distance field ϕ_0.

Simultaneously, by integrating the vertex coordinates $\mathbf{P}$ and vertex normals $\mathbf{N}$ of the input mesh, along with the previously constructed scalar fields ϕ_p and ϕ_0, the initial feature vector $x = [\mathbf{P}, \mathbf{N}, \phi_p, \phi_0]$ is formed. This feature vector serves as the input to the network, encapsulating geometric and constraint-related information necessary for the subsequent processing.

3.2. Network Architecture Design

Building upon the Physics-Informed Neural Network (PINN) framework, we introduce the Neural Implicit Curve Modeling on Meshes (NICMM) network. Unlike conventional PINN architectures that typically rely on straightforward multilayer perceptrons (MLPs), the NICMM network is specifically tailored to address problems defined on surface meshes in three-dimensional space. To effectively capture intricate geometric mappings and accelerate convergence, a hierarchical feature processing architecture is proposed, as illustrated in Fig. 2. Specifically, the NICMM architecture enhances the basic MLP structure with the following specialized modules:

- Efficient Channel Attention (ECA): This module heightens the network's responsiveness to features in the vicinity of control points by facilitating local cross-channel interactions. Through this mechanism, the network can more effectively prioritize relevant geometric features critical to curve design.
- Light Gate Linear Unit (Light GLU): By integrating a lightweight gating mechanism, this module significantly improves the network's feature selection capabilities while mitigating the parameter explosion issue commonly associated with traditional GLU structures. This design choice ensures computational efficiency without compromising the expressiveness of the network.

A comprehensive comparison of the contributions of each module to the overall efficiency of the proposed network is presented in Section 4.2.

3.3. Loss Function Design

The mathematical formulation of implicit curve constraints involves the combined effects of scalar field gradients and Laplace operators. Although traditional PINN methods (e.g. automatic differentiation and tangent plane projection) could be employed, discretization may introduce substantial errors that degrade the accuracy of the solution. To address this, we adopt the constraint construction approach proposed by Zhang et al. [38], utilizing the finite element method for discretization. Consequently, the loss function designed in this paper incorporates the following constraints: level-set constraint, interpolation constraint, smoothness constraint, and feature constraint.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sdf}} + \lambda_{\text{int}}\mathcal{L}_{\text{int}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}} + \lambda_{\text{fea}}\mathcal{L}_{\text{fea}} \tag{2}$$

Level Set Constraint: Drawing on the Eikonal equation, a numerically stable smooth energy functional is employed as a constraint:

$$\mathcal{L}_{\text{sdf}} = \int_M (|\nabla\phi|^2 - 1)^2 dM \tag{3}$$

This energy functional effectively constrains the geodesic distance field and ensures a reasonable distribution of the implicit curve on the discrete manifold.

Interpolation Constraint: The constraint ensures that the predicted values at the interpolation points are zero, thus ensuring that the control points are accurately fitted:

$$L_{\text{int}} = \sum_{i=1}^{|P|} \phi(p_i)^2. \tag{4}$$

Smoothness Constraint: The smoothness of the level set is constrained using Willmore energy:

$$\mathcal{L}_{\text{smooth}} = \int_\Gamma \kappa_g^2 dl = \int_M \kappa_g^2(\phi)\delta(\phi)|\nabla_\phi| dM, \tag{5}$$

where κ represents the geodesic curvature, and δ_ϕ is the Dirac function. Non-zero values of ϕ are weighted as zero to maintain continuity of the predicted field near the curve. To reduce the complexity of the solution, the geodesic distance field constraint $|\Delta\phi| = 1$ can be treated as a prior constraint. This simplifies the above formula and approximates the geodesic curvature κ using the Laplacian operator of the level set function ϕ leading to the following simplified form:

$$\mathcal{L}_{\text{smooth}} = \int_\Gamma \kappa_g^2 dl = \int_M \frac{1}{2}(\Delta\phi)^2\delta(\phi) dM \tag{6}$$

In some situations, to enhance control over curve behavior in specified regions, we also propose a feature-aware mechanism that integrates semantic or geometric features into the network, such as feature alignment and obstacle avoidance. These applications can be treated as feature-based guidance, where alignment regions are regarded as areas of attraction, and obstacle regions as areas of repulsion for the predicted curve.

To represent such feature-based guidance quantitatively, we introduce a scalar field that encodes the relative importance of different regions, i.e. the importance map. This field can be constructed manually by assigning values to user-specified feature vertices, or automatically based on mesh geometry. For example, a common strategy is to use geometric indicators such as the maximal principal curvature of the mesh vertices, as suggested in [34], to extract salient structural features such as sharp edges or ridges.

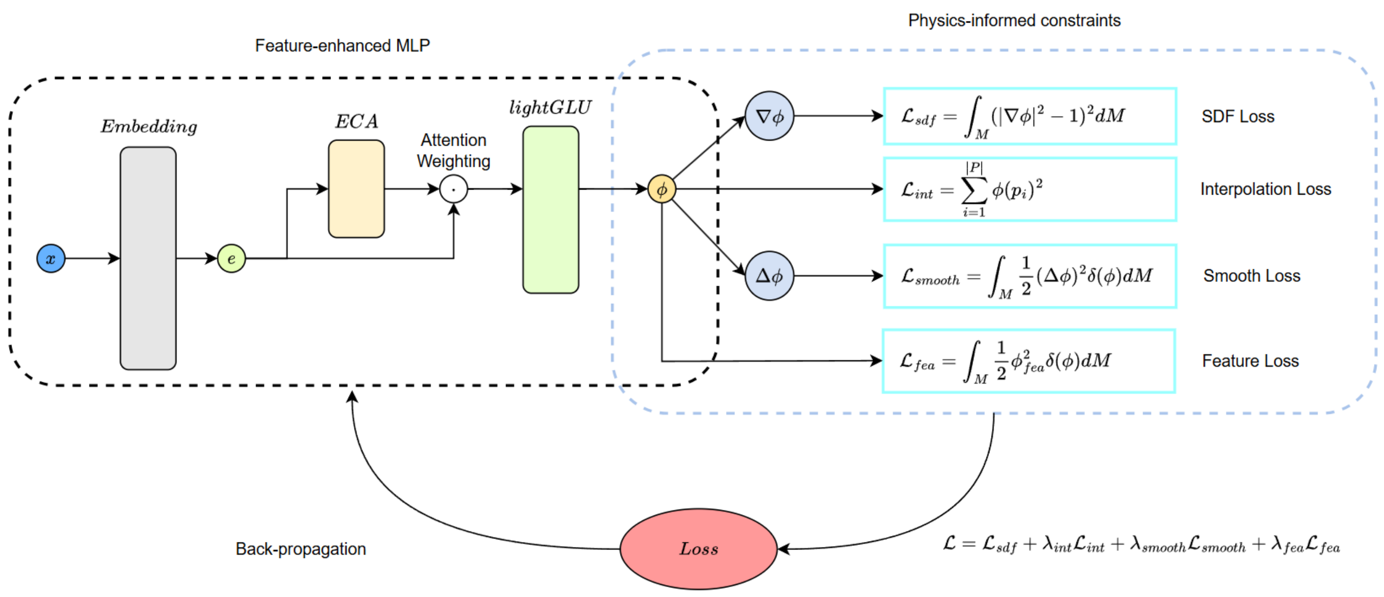
Feature-enhanced MLP
Physics-informed constraints
Embedding
ECA
Attention Weighting
lightGLU
x
e
ϕ
$\nabla\phi$
$\Delta\phi$
$\mathcal{L}_{sdf} = \int_M (|\nabla\phi|^2 - 1)^2 dM$
SDF Loss
$\mathcal{L}_{int} = \sum_{i=1}^{|P|} \phi(p_i)^2$
Interpolation Loss
$\mathcal{L}_{smooth} = \int_M \frac{1}{2}(\Delta\phi)^2\delta(\phi)dM$
Smooth Loss
$\mathcal{L}_{fea} = \int_M \frac{1}{2}\phi_{fea}^2\delta(\phi)dM$
Feature Loss
Back-propagation
Loss
$\mathcal{L} = \mathcal{L}_{sdf} + \lambda_{int}\mathcal{L}_{int} + \lambda_{smooth}\mathcal{L}_{smooth} + \lambda_{fea}\mathcal{L}_{fea}$

Fig. 2: NICMM architecture diagram

In our approach, given a user-defined set of feature vertices $\mathbf{P}_{fea}$ (which consists of both alignment and obstacle vertices), we construct a feature map ϕ_{fea} to quantify the feature intensity across the mesh. This map is generated by utilizing a Gaussian kernel to diffuse the influence of feature vertices to their adjacent vertices. Specifically, each feature vertex is endowed with a positive scalar weight, which serves as the standard deviation (σ) of its associated Gaussian kernel. The Gaussian value at an arbitrary mesh vertex is computed based on its signed distance function (SDF) from the feature vertex and the assigned σ. Following the calculation of these Gaussian responses, we assign negative signs to the values originating from alignment vertices and positive signs to those from obstacle vertices. The final value of the map at each mesh vertex is obtained by aggregating all the signed Gaussian responses from the feature vertices. Through this mechanism, the alignment vertices decrease the scalar values in their vicinity, whereas the obstacle vertices increase them, effectively encoding the distinct characteristics of different types of feature. Consequently, the degree of diffusion is adaptively regulated by the weights used as σ values in the kernels. All non-feature regions are initialized with a neutral baseline value. The resulting scalar field ϕ_{fea} is appended as an additional channel to the input features, forming a final feature vector: $x = [\mathbf{P}, \mathbf{N}, \phi_p, \phi_0, \phi_{fea}]$, where ϕ_{fea} serves as a unified region-aware descriptor, leading to a 9-dimensional input per vertex. To incorporate this feature-aware guidance into the training objective, we introduce a feature-aware loss term.

$$\mathcal{L}_{\text{fea}} = \int_M \phi_{fea}^2 \delta(\phi) dM. \tag{7}$$

The above loss of feature awareness encourages the zero level set (i.e., the predicted curve) to align with regions with lower ϕ_{fea} values while being pushed away from regions with higher values.

Classic PINN methods are inherently mesh-free and require the evaluation of loss at sampling points within the computational domain. To streamline computations and enhance the tractability of the problem, we employ a strategy where all mesh vertices are treated as sample points. Departing from the conventional PINN framework, we introduce a discretization scheme in which each loss function is computed over the Voronoi domain associated with the sample points. This approach ensures a more robust and precise numerical integration, effectively improving the accuracy and stability of the overall optimization process. Building on this foundation, the level set loss, smoothness loss and feature aware loss terms of the proposed network are discretized as follows:

$$\begin{aligned}
E_{\text{sdf}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \tfrac{1}{2}(|\nabla\phi(c_t)|^2 - 1)^2 \\
E_{\text{smooth}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \tfrac{1}{2}(\Delta\phi(c_t))^2 \cdot G_\sigma(\phi(c_t)) \\
E_{\text{fea}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \tfrac{1}{2}(\phi_{fea})^2 \cdot G_\sigma(\phi(c_t))
\end{aligned} \tag{8}$$

where $\mathcal{N}(v)$ represents the set of neighboring triangles of vertex v, c_t is the centroid of triangle t, $w_{i,t}$ is the area of the Voronoi domain associated to vertex i in triangle t, typically set as $|t|/3$, is the Gaussian weight based on the distance value $\phi(c_t)$, used to approximate the Dirac delta function.

4. Experimental Results

All experiments were carried out on the Ubuntu 22.04 operating system within a high performance computing environment. The hardware configuration includes an Intel(R) CoreTM i9-14900K CPU with a base frequency of 3.20 GHz, 64 GB of system memory, and a NVIDIA GeForce GTX 4090 graphics card with 24 GB of dedicated VRAM for accelerated graphics processing. The CUDA parallel computing platform (version 12.6) was utilized to efficiently leverage GPU's computational capabilities. For software implementation, the experiments were developed using the PyTorch 2.3.1 deep learning framework, which offered a flexible and efficient environment for network development and training.

The experiments in this paper are conducted primarily on mesh models from the SHREC16 dataset provided by MeshCNN[7]. This dataset is chosen due to its comprehensive nature and wide acceptance within the academic community, which offers a diverse range of mesh structures that are representative of various real-world scenarios. It encompasses meshes with different geometries, topologies, and levels of complexity, thereby providing a robust and challenging testing ground for the proposed methods. However, some meshes within the dataset exhibit a relatively sparse vertex distribution. To address this issue and ensure more accurate and reliable experimental results, we apply subdivisions for these meshes.

4.1. NICMM training strategy

We employ NICMM to conduct curve design experiments on various mesh models. To improve the efficiency of curve generation, we partitioned the NICMM training procedure into two stages.

The initial phase is the pretraining stage, during which a batch of control points $\{(\mathbf{P}_i\}_{i=1}^{N}$ is randomly sampled from the mesh surface to conduct preliminary training. The optimization objective for this stage is formulated as:

$$\Theta^* = arg\min_{\Theta} \sum_{i}^{N} \mathcal{L}_{\text{total}}(\phi_{\Theta}(\mathbf{P}_i)), \tag{9}$$

where, $\phi_{\Theta}(\mathcal{P}_i)$ is the predicted result of the network at the control point $\mathcal{P}_i$, with the aim of acquiring a set of network parameters Θ^*. This stage necessitates only a brief pre-training duration, which notably curtails the time required for the subsequent stage.

Specifically, for each network under evaluation, we randomly sample 1000 sets of control points from its vertex set, each set comprising approximately 3 to 5 vertices. These control points are subsequently transformed into initial constraint features, which, along with the mesh's vertex coordinates and normal vectors, serve as input features for training NICMM. The training process uses the AdamW optimizer with an initial learning rate of 0.01.

The second stage is the prediction stage, where the learning rate of the AdamW optimizer is adjusted to 0.001. The mesh and user-specified constraint conditions are input into the network pre-trained in the first stage for further optimization. Experiments evaluations reveal that the NICMM model achieves a remarkable acceleration in the prediction stage convergence by undergoing merely around 10 epochs of pre-training, which approximately takes 2 minutes. The rapid convergence enabled by the pre-training phase allows the NICMM model to swiftly adapt to the characteristics of the dataset, thereby facilitating more accurate and timely predictions.

To assess the impact of varying pretraining durations on accelerating the prediction process, we selected a mesh model from the dataset and randomly sampled 1000 control point sets for training across different durations, followed by 50 random test sets. Using each pre-trained network, we performed the prediction stage 30 times and recorded the average loss of the test set, as presented in Table 1:

Table 1: Comparison of average loss for varying prediction durations

Pre-training Duration	Average Loss
No Pre-training	57.6
2 min	19.9
10 min	14.4
30 min	13.0
10 h	**12.5**

Note: Bold indicates the best value.

In the prediction stage, the network is optimized by minimizing curvature-related energy terms across the entire mesh. These energy terms include a gradient norm-based smoothing term and a Laplace operator-based bending term. However, due to the nature of curve generation tasks, the predicted results only need to exhibit desired geometric properties within the narrow-band region adjacent to the zero-level set. Continuing to optimize these energy terms in regions far from the zero-level set not only marginally improves the curve quality but also wastes computational resources and may induce numerical oscillations.

To evaluate whether a pre-trained network obtained from a single mesh can still perform well on other meshes, a two-stage training strategy is designed. Stage 1 involves sampling 1000 sets of control points on a given mesh for pretraining, allowing the network to learn generalizable geometric features. Stage 2 selects 20 meshes with diverse connective and geometry from the dataset, with 10 control point sets sampled per mesh to form a validation dataset of 200 test samples. All test samples are initialized with the Stage 1 pre-trained network and undergo 50 iterations of constraint optimization to adapt to new mesh geometries.

As shown in Table 2, the network with pretraining consistently achieves lower loss values compared to the model trained from scratch, reflecting faster convergence and better optimization quality. This highlights the strong generalizability of the proposed pretraining strategy across meshes with varying geometric and topological characteristics.

Table 2: Comparison of network with and without pretraining

Pretraining Duration	Average Loss
Without Pretraining	15.298
Pretrained for 30 min	10.996

To prevent the network from engaging in ineffective training during later stages, we devised an early stopping strategy to assess whether the curve predicted by the current network satisfies the generation requirements, thus terminating the prediction process prematurely. Specifically, the following steps are taken at the end of each epoch.

1. Extract zero-level set curve: Extract the contour with a level set value of 0 from the network-predicted level set function, which serves as the current predicted curve.
2. Evaluate the metric of narrow-band energy: Compute the average values of the smoothing and curvature energy terms exclusively within a narrow band (with a width of three times the mesh's average edge length) surrounding the curve.
3. Determine the convergence condition: If the narrow-band energy exhibits minimal fluctuations over several consecutive epochs or the energy drops below a predefined threshold, the curve is deemed to satisfy the geometric constraints, prompting the termination of the training.

To validate the impact of the early stop strategy on training efficiency and results, we iterated the training process for the same data set multiple times and recorded the evolution of the loss function, as illustrated in Fig. 3. The loss function notably plateaus after 200 iterations.

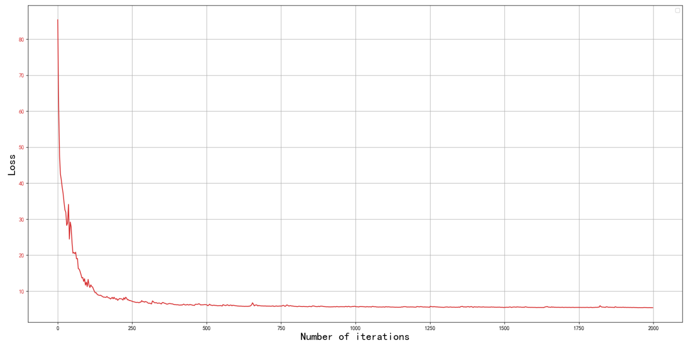

Fig. 3: Change of loss with respect to the number of iterations.

To evaluate the impact of the early stop strategy on training efficiency and curve quality, we designed the following comparative experiment. After the predicted curve first satisfies the early stop criterion (that is, the differential energy within the narrow band region falls below a predefined threshold), we recorded the curve prediction results in three distinct settings, as depicted in Fig. 4. The curves in the figure correspond to different iteration counts, and visual inspection reveals that the predictions obtained after varying numbers of iterations are nearly indistinguishable. Their geometric and topological properties remain stable and exhibit only minor discrepancies in local details. However, training durations vary significantly, indicating that excessive iterations yield little improvement in result while consuming computational resources and potentially introducing numerical instability.

4.2. Ablation Study

To validate the contribution of each module to the NICMM network architecture for curve prediction tasks, we performed an ablation study by separately removing the ECA module and the LightGLU module. The performance of the full network was compared with variants under identical conditions: 40 test datasets, 30 minutes of pre-training time, and 50 rapid convergence iterations. The results are tabulated in Table 3:

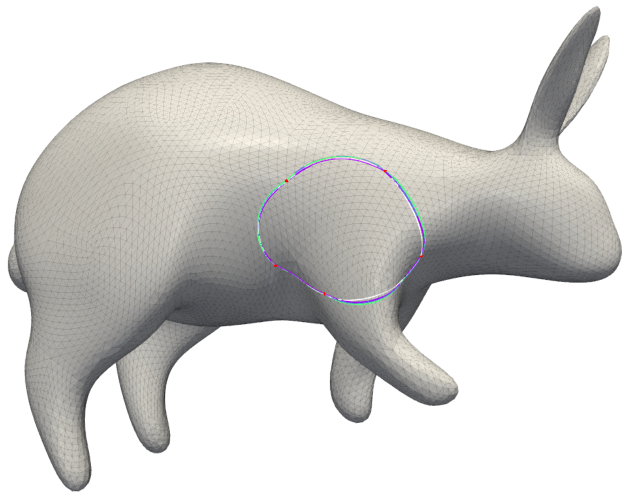

Fig. 4: The figure shows four performance curves with different prediction frequencies: blue (41 iterations, meeting early-stopping criteria), orange (100 iterations), green (200 iterations), and purple (500 iterations)

Table 3: Convergence efficiency across model variants. Bold indicates optimal values. "w/o" = without.

Network	Time (ms)	Loss
NICMM	391	**8.84**
w/o ECA	290	13.34
w/o LightGLU	301	14.89
w/o Both	**262**	19.54

To further investigate the influence of diverse initial feature inputs on curve prediction tasks, we performed an ablation study on the input feature construction component. Specifically, we systematically removed the vertex normal vectors, initial interpolation scalar field, and initial level set features to quantify the contribution of each feature to the network's convergence dynamics. All experiments used the early stopping strategy and tracked the iterations required for different combinations of features to satisfy the stopping criteria. The evaluations were conducted under consistent conditions: 10 test samples and 2 minutes of pre-training. The results are presented in Table 4:

Table 4: Comparison of prediction with Different Initial Features. Bold indicates the best performance. "w/o" = without.

Initial Feature Combination	Average Iterations
Full Features	**32.0**
w/o Vertex Normals	35.4
w/o Interpolation Scalar Field	75.4
w/o Initial Level Set	77.2

As shown in Table 4, the network achieves the early stop criterion with minimal iterations with complete input features. The removal of any feature subset consistently prolongs training iterations, with the most pronounced effect evident when the initial level set is excluded; this leads to an average increase of approximately 45.2 steps. Although removing vertex normal vectors also introduces a convergence delay, its impact is comparatively minor.

4.3. Parameters

In the loss function (Eq. (2)), the weights are employed to balance the influence of different energy terms, and their values can exert a certain impact on the results. In this experiment, the default recommended values are set as follows: $\lambda_{sdf} = 1, \lambda_{int} = 5000$, $\lambda_{smooth} = 500$ and $\lambda_{fea} = 0$ (Notably, since the interpolation constraint is regarded as a hard constraint, it is assigned a large fixed weight. The weight of feature constraint λ_{fea} is set to zero by default, as feature-aware guidance is not considered here. Details on feature constraints can be found in Section 4.7). Fig. 5 shows an ablation study on the remaining two energy weights. In figures (a) and (b), the white, blue, and green curves, respectively, represent the predicted results as the weights of the level set term and the smoothness term increase. It can be observed that the predicted curves under different settings of λ_{sdf} and λ_{smooth}, remain quite similar, suggesting that the overall shape of the predicted curves changes only slightly and maintains stability. The performance of the network is generally consistent with that of traditional variational methods. This shows that the network is robust to weight settings within a reasonable range and can consistently generate the target curve without significant performance fluctuations due to hyperparameter tuning.

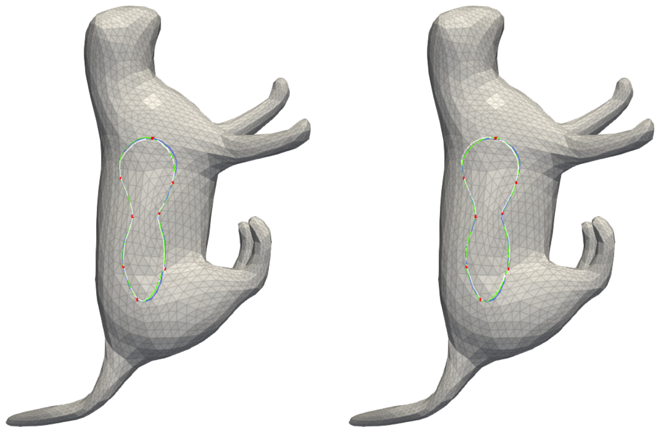
(a) $\lambda_{sdf} = 0.1, 1, 10$ (b) $\lambda_{smooth} = 50, 500, 5000$

Fig. 5: Performance of NICMM under different initialization weights.

4.4. Sensitivity of different initial values

In the curve generation process governed by control point constraints, the NICMM network leverages the initial scalar field ϕ_0—derived via the Heat Method from the closed-loop curve formed by the control points as input. Consequently, the methodology for constructing the initial closed loop may influence the final predicted curve.

To evaluate NICMM's sensitivity to the initial scalar field, we designed the following experiment: Despite maintaining fixed control point constraints, we manually constructed multiple initial closed-loop curves with distinct geometries, resulting in diverse initial interpolation scalar fields ϕ_p and corresponding initial level sets ϕ_0. These initial fields were then fed into the trained NICMM network, and variations in the final predicted curves were systematically analyzed. Fig. 6 presents a comparison of the predicted curves generated from different initial closed loops with identical control points. The experimental results show that, as the geometry of the initial closed loop varies, the curves generated by NICMM also differ. However, these differences appear mainly in local geometric details, whereas the overall shape and topological structure remain stable. This indicates that although the method exhibits some sensitivity to the initial closed loop, it does not rely on strictly constructing "optimal initial values" in practical applications and demonstrates strong convergence robustness.

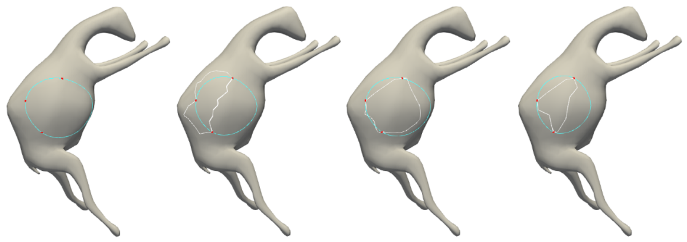
Fig. 6: The white curves in the figure represent manually constructed initial closed loops, while the blue curves denote the predicted results by the network. The leftmost image shows a case without an initial loop, using only interpolation points to generate the signed distance field.

4.5. Comparisons

Given that NICMM's physical constraints are consistent with those of traditional optimization-based methods [38], both of which rely on discrete differential operators for computation. Thus, a comparative analysis was conducted between the two approaches. A mesh model with about 4000 vertices and 8000 faces was selected and a common set of control points was used to generate curves via the traditional implicit method and the NICMM network. The results are shown in Fig. 7, where the white curve represents the results of the traditional method and the blue curve denotes the prediction results of NICMM. Visual inspection indicates that the two methods produce generally similar results, although there are subtle discrepancies. These differences are likely attributed to variations in initial closed loops, which cause the solutions to converge to different local minima, thereby resulting in slightly divergent curves.

To quantitatively evaluate the quality of the curves produced by the two methods, i.e. smoothness, we adopt geodesic curvature as the assessment metric. Specifically, the zero level set is extracted from the implicit scalar field generated by both NICMM and the variational method [38] and subsequently discretized into polyline representations. Using these representations, the geodesic curvature κ_g is calculated at each vertex.

$$\kappa_g = \frac{2sin\theta_i}{|\hat{r}_{i+1} - \hat{r}_{i-1}|}, \tag{10}$$

where r_{i-1}, r_i, r_{i-1} represent three adjacent vertices on the curve, and their projections onto the tangent plane at the point r_i are

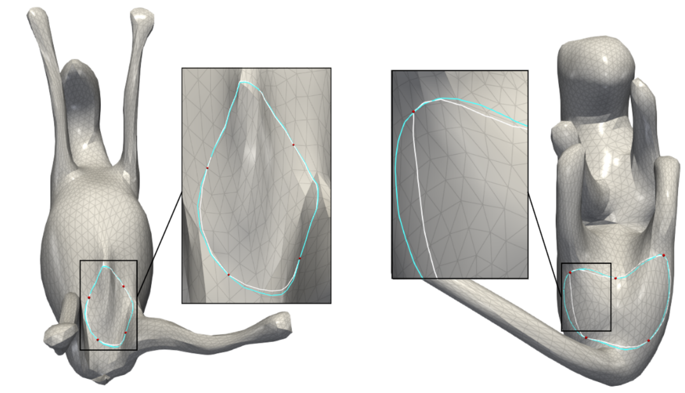

Fig. 7: Comparison of variational methods.

denoted as $\hat{r}_{i-1}$, $\hat{r}_i$, $\hat{r}_{i-1}$, θ_i is the angle between $\hat{r}_i - \hat{r}_{i-1}$ and $\hat{r}_{i+1} - \hat{r}_i$. Fig. 8 shows the geodesic curvature distribution of the curves generated by the two methods. Geodesic curvature values are visualized as a weighted histogram, where the weight of each bin is determined by the length of the corresponding curve segment. The experimental findings demonstrate that the curves generated by the proposed method (blue) and the variational approach (green) exhibit comparable distributions of geodesic curvature, suggesting a high degree of similarity in their geometric characteristics.

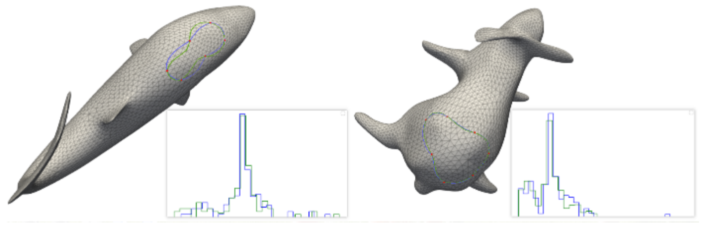

Fig. 8: Comparison of geodesic curvature across different methods.

To assess the efficiency of our method relative to the traditional variational approach, we designed the following experiment: 40 sets of control point data were randomly selected from the dataset, and curve generation was performed for each set using both the NICMM and the variational method. To mitigate random error, each experiment was repeated three times, with the average runtime recorded as the final result. Both methods were executed on the same hardware platform under identical convergence criteria (that is, the loss variation over the last 10 iterations was less than 10^{-5}). The average runtime and standard deviation are reported in Table 5.

Table 5: Comparison of runtime (averaged over three runs) between NICMM (ML) and the variational method (VM) on the test data.

Method	Avg (ms)	Std (ms)	Min (ms)	Max (ms)
ML	398	**62**	262	697
VM	**102**	106	**24**	**538**

Note that although NICMM can utilize GPU acceleration to facilitate efficient computation, the requirement of performing backpropagation through the entire network at each iteration leads to lower computational efficiency compared to the traditional variational method, which operates on the CPU.

To further validate the advantages of the proposed NICMM framework in curve modeling, we perform a comparative analysis against B-surf [20], a state-of-the-art explicit method designed to generate spline curves on meshes. The results, visualized in Fig. 9, show fundamental differences between the two approaches. B-surf employs a piecewise Bézier segment construction strategy, which needs manual specification of control points and knot vectors for each individual segment. This process requires user intervention to ensure continuity across segment boundaries, often resulting in a labor intensive workflow. In contrast, NICMM utilizes implicit representations to automatically generate smooth curves that satisfy interpolation constraints using only user-specified control points. By integrating geometric fairness principles directly into the neural network architecture, such as incorporating curvature regularization terms in the loss function, NICMM inherently produces high-quality fair curves with globally consistent smoothness. This eliminates the need for manual segmenting and reduces reliance on experience. Notably, while B-surf may generate curves with localized high-curvature regions, NICMM ensures a more uniform curvature distribution. These results highlight NICMM's capability to streamline the curve design process through automation while enhancing curve quality.

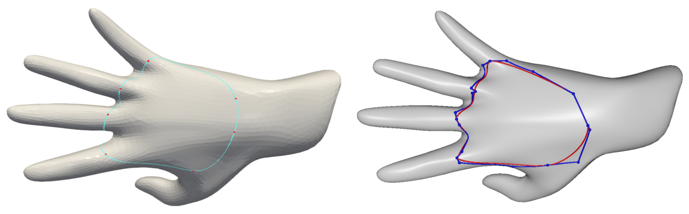

Fig. 9: Comparison with explicit methods. Left: Smooth implicit curve generated by NICMM from user-specified control points, exhibiting geometric fairness with uniform curvature distribution. Right: Explicit spline curve produced by B-surf under identical input points, with overlaid Bézier control points (blue) and knot vectors (red) highlighting manual segment and continuity adjustments.

4.6. Robustness analysis

To evaluate the robustness of the network under varying mesh quality conditions, this study synthesizes a set of low-quality meshes based on the subdivided SHREC16 dataset using a geometric perturbation method that introduces near-degenerate elements. This method preserves the mesh topology while deliberately disturbing the positions of certain vertices, thereby generating elongated triangular faces and significantly degrading overall mesh quality.

Specifically, to introduce geometric perturbations, we randomly select one triangle from the set of mesh faces F for every n triangle. For each selected triangle, we randomly choose one vertex and displace it along the perpendicular direction of the plane defined by the opposite edge. The magnitude of displacement is determined as a random fraction of the length of the opposite edge, constrained within the interval $[s_{min}, s_{max}]$. This approach ensures a controlled and consistent level of geometric distortion across the mesh. To safeguard against excessive vertex movement and self - intersections, a KD - Tree - based collision detection mechanism is employed prior to each perturbation operation. This collision detection step filters out any vertex displacements that would cause the vertex to come within an unacceptable proximity of neighboring vertices. Following the perturbation, the modified vertex is projected back onto the surface of its associated triangle. Subsequently, the internal angles of the triangle are computed, and if any of these angles fall below a pre - defined threshold (such as 1°), the perturbation is discarded, ensuring that the mesh quality remains within an acceptable range. The resulting meshes visually retain the overall geometry shape, but their local quality is significantly degraded. These meshes serve as effective test cases for evaluating the network's performance under irregular boundaries and extreme aspect ratios. Fig. 10 shows an example of a perturbed low quality mesh. When designing smooth curves on surface

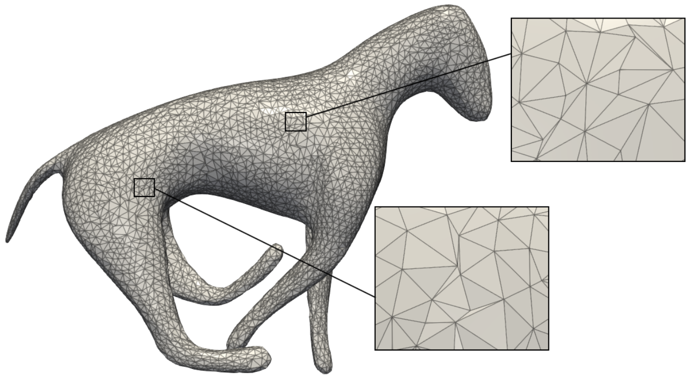
Fig. 10: Schematic diagram of low-quality mesh.The red bounding boxes highlight regions containing nearly degenerate patches.

meshes using traditional numerical optimization methods (e.g., gradient descent or Newton's method), the robustness of these approaches is highly dependent on the quality of the mesh. In the presence of elongated or near-degenerate triangles, the condition number of the resulting nonlinear system can degrade drastically, leading to slow convergence, oscillatory behavior, or even complete failure. Although our approach employs a discretization scheme similar to traditional differential operators, it uses physics-informed machine learning to mitigate such numerical instabilities. By implementing single-sample repeated training during the rapid convergence phase, the proposed network implicitly learns the local geometric features of the underlying mesh. This mechanism alleviates the adverse effects of local numerical instability, enabling the network to maintain robust stability even when processing low-quality meshes with extreme element aspect ratios or topological imperfections.

The experimental results in Fig. 11 demonstrate the exceptional robustness of the proposed method compared to traditional optimization techniques when confronted with nearly degraded mesh conditions. NICMM exhibits remarkable resilience, consistently generating high - fidelity curves that adhere to interpolation and smoothness requirements, even in the presence of suboptimal mesh quality. In contrast, the variational approach succumbs to numerical instabilities in regions with mesh irregularities. These instabilities manifest as numerical divergence, leading to the emergence of severe visual artifacts and, in some cases, the complete failure to meet the imposed geometric constraints. This stark contrast underscores the significant advantage of NICMM in handling challenging mesh scenarios, solidifying its superiority in practical applications where mesh quality may vary widely.

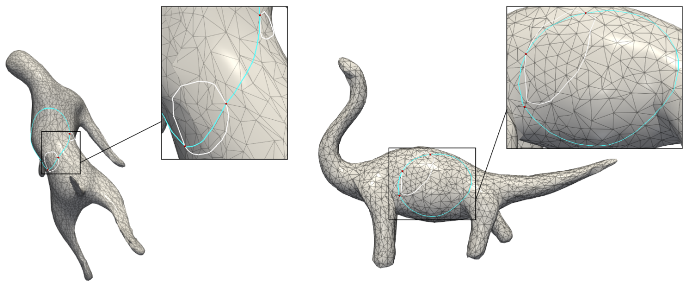
Fig. 11: Comparison results on low-quality meshes.NICMM (blue) vs. traditional variational method [38] (white).

To assess the stability of NICMM on the surface with different discretizations, we designed two experiments. Firstly, we conducted geometric perturbations for a high-quality mesh and generated multiple mesh variants with nearly degenerate elements. With a fixed set of control points, our approach produces corresponding curves as shown in the upper row of Fig. 12, where all curves successfully interpolate control points while maintaining nearly identical shapes, even when mesh quality is severely degraded. This outcome underscores the robust resilience of the approach to geometric noise and topological imperfections in the input mesh.

Secondly, we selected an additional mesh from the SHREC16 dataset and applied successive levels of Loop subdivision to generate meshes with varying resolutions (coarse, medium, and fine). Using the same set of control points for meshes with various resolutions, we evaluated the consistency of predicted curves. As shown in the bottom row of Fig. 12, NICMM delivers stable predictions and faithfully adheres to control point constraints across all mesh resolutions. Notably, the method exhibits remarkable insensitivity to mesh vertex density, maintaining consistent geometric fidelity, whether applied to low-resolution meshes (≈1k vertices) or high-fidelity fine meshes (≈64k vertices). This demonstrates NICMM's robust generalization capability across diverse geometric representations, ensuring reliable performance in both sparse and dense mesh environments.

4.7. Feature-aware curve design

In the experiment depicted in Fig. 13, we designated a set of alignment vertices to validate the ability to align features. The right image shows that the predicted curve tightly conforms to the specified region, providing empirical evidence of the feature-aware design's effectiveness in guiding curve generation according to user-specified constraints.

Fig. 14 presents the result of integrating obstacle constraints. By assigning larger values to obstacle regions in the feature

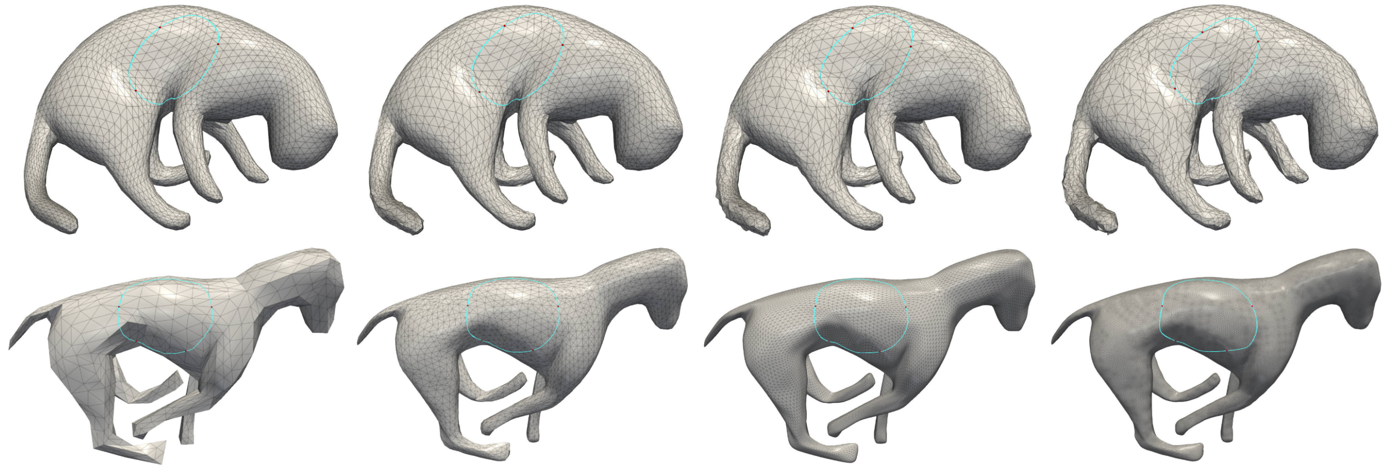
Fig. 12: Illustration of the robustness of NICMM under varying mesh qualities and resolutions. The top row shows mesh surfaces with different levels of geometric perturbations applied to a high-quality base mesh. The bottom row presents the corresponding curves predicted by NICMM on meshes with varying degrees of subdivision, using the same set of control points.

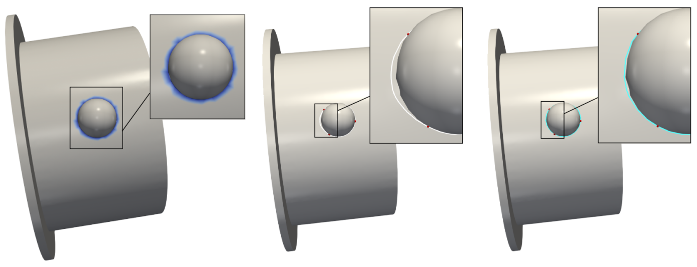
Fig. 13: From left to right: input with alignment region (blue), result without guidance, and result with feature alignment.

field, the curve successfully avoids undesired areas, demonstrating the unified treatment of different types of guidance.

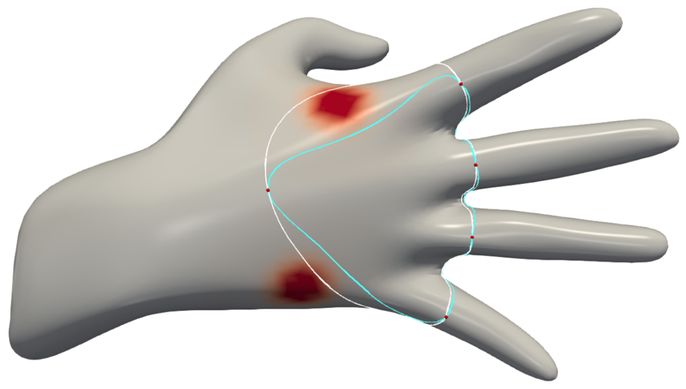
Fig. 14: In the figure, the white and blue curves represent the network's predictions for the same control points without and with obstacle constraints.

To evaluate the generalizability of NICMM across diverse geometric representations and topological configurations, we conducted comprehensive experiments on multiple benchmark mesh datasets. Specifically, we selected a representative subset from the Thingi10K repository, which encompasses a broad spectrum of real-world objects with varying complexity, including organic shapes, mechanical parts, and architectural models. As shown in the visual gallery presented in Fig. 15, the network demonstrated remarkable stability and consistency in generating high-quality curves across this heterogeneous dataset. The results indicate that NICMM can effectively handle meshes with intricate geometries, non-uniform sampling densities, and topological variations.

5. Conclusion

We propose NICMM, a novel curve modeling network grounded in physics-informed neural networks (PINNs), by integrating deep learning techniques with traditional implicit curve design methodologies. This innovative network takes triangular meshes and control points as inputs and computes a level set function that adheres to multiple geometric constraints. Consequently, it enables the generation of curves that reside precisely on surface meshes. To optimize the performance of the network, we devise a two-stage training strategy. In the initial stage, a concise pretraining process is carried out using control points sampled from mesh vertices. Subsequently, in the second stage, the network achieves fast convergence for user-defined control points through optimization.

The experimental findings unequivocally demonstrate that the NICMM outshines in both robustness and generalization capabilities. Significantly, it maintains consistent performance across meshes with disparate discretization schemes, highlighting its adaptability to various geometric representations. Notably, even when applied to low-quality meshes marred by elongated faces or acutely small angles, NICMM adeptly generates smooth and geometrically valid curves, thereby underscoring its exceptional resilience against mesh degradation. Moreover, the model exhibits remarkable insensitivity to the initial level set construction. Irrespective of the initialization method employed, the network reliably converges to analogous predictions, effectively eliminating the need for extensive manual fine-tuning. This characteristic not only streamlines the computational process but also enhances the practical utility

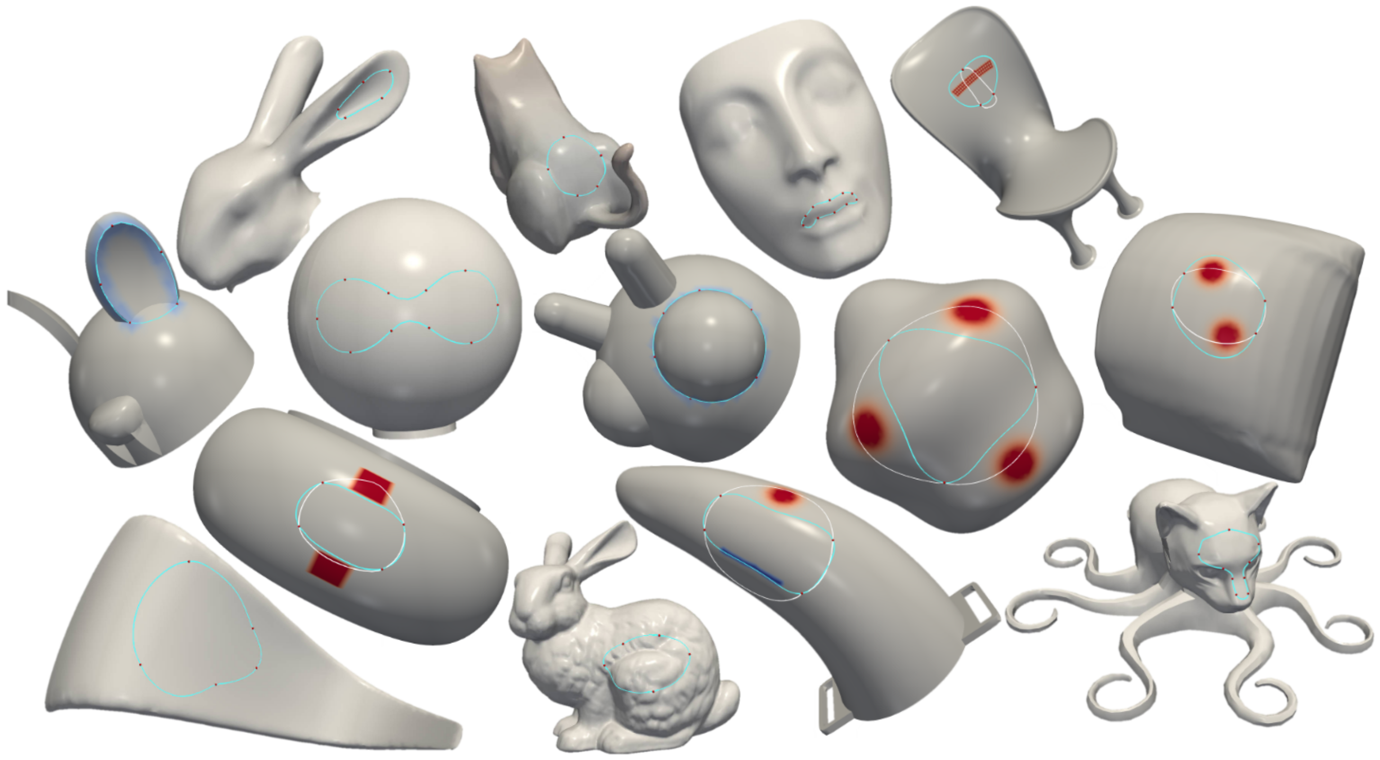

Fig. 15: The gallery of curve design results by NICMM on various meshes. The red regions marked on the models indicate obstacle areas, and the blue regions indicate feature alignment areas. The white curves are generated without feature constraints.

of NICMM, making it a highly versatile and user-friendly approach for curve generation tasks.

Despite these advances, the proposed approach has some limitations, and several critical challenges remain ripe for further investigation. Firstly, although the current network demonstrates promising capabilities, its prediction efficiency has yet to achieve a significant edge over traditional numerical approaches. To address this, future research endeavors will focus on exploring more efficient and streamlined network architectures, aiming to facilitate swift and direct curve prediction, thereby bridging the performance gap. Secondly, the network's inability to exert explicit control over the topologies of generated curves poses a notable drawback. As depicted in Fig. 16, this limitation may result in the emergence of unanticipated structures, such as multiple loops, which can deviate from the intended design outcomes. To overcome this hurdle, subsequent studies will seek to incorporate topological constraint mechanisms. By doing so, the approach will not only enhance the controllability and reliability of curve generation, but also ensure that the produced curves align more closely with user expectations, leading to more predictable and user-oriented designs.

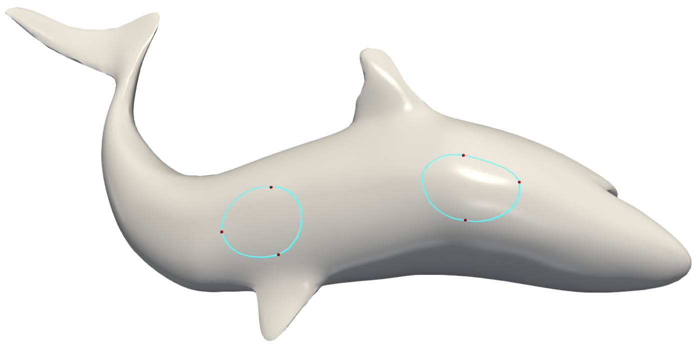

Fig. 16: Predicted curve contains multiple loops due to the lack of topological control.

References

[1] Chunlin Wu and Xuecheng Tai. A Level Set Formulation of Geodesic Curvature Flow on Simplicial Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):647–662, July 2010.

[2] Francisco Sahli Costabal, Simone Pezzuto, and Paris Perdikaris. δ-pinns: Physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, 127:107324, 2024.

[3] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Communications of the ACM*, 60(11):90–99, October 2017.

[4] Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

[5] A. Gehre, M. Bronstein, L. Kobbelt, and J. Solomon. Interactive Curve Constrained Functional Maps. *Computer Graphics Forum*, 37(5):1–12, August 2018.

[6] Frederic Gibou, Ronald Fedkiw, and Stanley Osher. A review of level-set methods and some recent applications. *Journal of Computational Physics*, 353:82–109, January 2018.

[7] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A network with an edge. *ACM Transactions on Graphics*, 38(4):1–12, August 2019. arXiv:1809.05910 [cs].

[8] Michael Hofer and Helmut Pottmann. Energy-Minimizing Splines in Manifolds.

[9] Ping Hu, Bing Shuai, Jun Liu, and Gang Wang. Deep Level Sets for

Salient Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 540–549, Honolulu, HI, July 2017. IEEE.

[10] Zhongping Ji, Ligang Liu, Zhonggui Chen, and Guojin Wang. Easy mesh cutting. In *Computer Graphics Forum*, volume 25, pages 283–291. Wiley Online Library, 2006.

[11] Yao Jin, Dan Song, Tongtong Wang, Jin Huang, Ying Song, and Lili He. A shell space constrained approach for curve design on surface meshes. *Computer-Aided Design*, 113:24–34, August 2019.

[12] Moonryul Jung and Haengkang Kim. Snaking across 3d meshes. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 87–93. IEEE, 2004.

[13] Lotan Kaplansky and Ayellet Tal. Mesh Segmentation Refinement. *Computer Graphics Forum*, 28(7):1995–2003, October 2009.

[14] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, May 2021.

[15] Ali Lasemi, Deyi Xue, and Peihua Gu. Recent development in cnc machining of freeform surfaces: A state-of-the-art review. *Computer-Aided Design*, 42(7):641–654, 2010.

[16] Kai Lawonn, Rocco Gasteiger, Christian Rössl, and Bernhard Preim. Adaptive and robust curve smoothing on surface meshes. *Computers & Graphics*, 40:22–35, May 2014.

[17] Yunjin Lee and Seungyong Lee. Geometric Snakes for Triangular Meshes.

[18] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, July 2005.

[19] Zheng Liu, Huayan Zhang, and Chunlin Wu. On Geodesic Curvature Flow with Level Set Formulation Over Triangulated Surfaces. *Journal of Scientific Computing*, 70(2):631–661, February 2017.

[20] Claudio Mancinelli, Giacomo Nazzaro, Fabio Pellacini, and Enrico Puppo. b/surf: Interactive bézier splines on surface meshes. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3419–3435, 2023.

[21] Claudio Mancinelli and Enrico Puppo. Straightedge and Compass Constructions on Surfaces. *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, page 10 pages, 2021. Artwork Size: 10 pages ISBN: 9783038681656 Publisher: The Eurographics Association Version Number: 001-010.

[22] Claudio Mancinelli and Enrico Puppo. Vector graphics on surfaces using straightedge and compass constructions. *Computers & Graphics*, 105:46–56, 2022.

[23] Claudio Mancinelli and Enrico Puppo. Computing the Riemannian center of mass on meshes. *Computer Aided Geometric Design*, 103:102203, June 2023.

[24] Claudio Mancinelli and Enrico Puppo. Splines on manifolds: A survey. *Computer Aided Geometric Design*, 112:102349, July 2024.

[25] D. Marin, F. Maggioli, S. Melzi, S. Ohrhallinger, and M. Wimmer. Reconstructing Curves from Sparse Samples on Riemannian Manifolds. *Computer Graphics Forum*, 43(5):e15136, August 2024.

[26] Dimas Martínez Morera, Paulo Cezar Carvalho, and Luiz Velho. Modeling on triangulations with geodesic curves. *The Visual Computer*, 24(12):1025–1037, December 2008.

[27] Daniele Panozzo, Ilya Baran, Olga Diamanti, and Olga Sorkine-Hornung. Weighted averages on surfaces. *ACM Transactions on Graphics*, 32(4):1–12, July 2013.

[28] M. Pawellek, C. Rössl, and K. Lawonn. Distance-Based Smoothing of Curves on Surface Meshes. *Computer Graphics Forum*, 43(5):e15135, August 2024.

[29] Helmut Pottmann and Michael Hofer. A variational approach to spline curves on surfaces. *Computer Aided Geometric Design*, 22(7):693–709, October 2005.

[30] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[31] Johannes Wallner and Helmut Pottmann. Intrinsic subdivision with smooth limits for graphics and animation. *ACM Transactions on Graphics*, 25(2):356–374, April 2006.

[32] Zian Wang, David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Object Instance Annotation With Deep Extreme Level Set Evolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7492–7500, Long Beach, CA, USA, June 2019. IEEE.

[33] Shiqing Xin, Pengfei Wang, Rui Xu, Dongming Yan, Shuangmin Chen, Wenping Wang, Caiming Zhang, and Changhe Tu. SurfaceVoronoi: Efficiently Computing Voronoi Diagrams Over Mesh Surfaces with Arbitrary Distance Solvers. *ACM Transactions on Graphics*, 41(6):1–12, December 2022.

[34] Rongyan Xu, Yao Jin, Huaxiong Zhang, Yun Zhang, Yu-kun Lai, Zhe Zhu, and Fang-Lue Zhang. A variational approach for feature-aware B-spline curve design on surface meshes. *The Visual Computer*, 39(8):3767–3781, August 2023.

[35] Cem Yuksel. A Class of C^2 Interpolating Splines. *ACM Transactions on Graphics*, 39(5):1–14, October 2020.

[36] Stefan Zachow, Evgeny Gladilin, Robert Sader, and Hans-Florian Zeilhofer. Draw and cut: intuitive 3D osteotomy planning on polygonal bone models. *International Congress Series*, 1256:362–369, June 2003.

[37] Juyong Zhang, Chunlin Wu, Jianfei Cai, Jianmin Zheng, and Xue-cheng Tai. Mesh Snapping: Robust Interactive Mesh Cutting Using Fast Geodesic Curvature Flow. *Computer Graphics Forum*, 29(2):517–526, May 2010.

[38] Xiaohu Zhang, Shuang Wu, Jiong Chen, Yao Jin, Hujun Bao, and Jin Huang. Versatile curve design by level set with quadratic convergence. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–10, 2024.

[39] Haikuan Zhu, Juan Cao, Yanyang Xiao, Zhonggui Chen, Zichun Zhong, and Yongjie Jessica Zhang. TCB-spline-based Image Vectorization. *ACM Transactions on Graphics*, 41(3):1–17, June 2022.