Computers & Graphics 132 (2025) 104351

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on CAD/Graphics 2025

Neural implicit curve: A robust curve modeling approach on surface meshes☆

Jintong Wang [a,b], Qi Zhang [a], Yun Zhang [c], Yao Jin [a,b,*], Huaxiong Zhang [a,**], Lili He [a,b,**]

[a] Zhejiang Sci-Tech University, Hangzhou 310018, China
[b] Zhejiang Provincial Innovation Center of Advanced Textile Technology, Shaoxing, 312000, China
[c] Communication University of Zhejiang, Hangzhou 310051, China

ARTICLE INFO

Keywords:
Implicit curves
Surface meshes
Robust modeling
Neural network

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 Neural Implicit Method with geometric constraints for robust curve design. NICMM leverages physics-driven loss functions to encode positional, smoothness, and other customized constraints, alleviates numerical instabilities and inaccuracies arising from low-quality meshes, such as convergence failures. The framework incorporates specialized modules (e.g., Efficient Channel Attention and Light GLU) to enhance feature extraction and computational efficiency and introduces a two-stage training strategy combining pre-training with rapid convergence optimization. Extensive experiments on the SHREC16 dataset demonstrate that NICMM has proven its mettle by outperforming traditional variational approaches in robustness. In the face of highly degraded meshes replete with elongated and near-degenerate elements, NICMM not only excels in generating high-fidelity curves but also maintains computational efficiency comparable to existing variational method, thereby showcasing its remarkable balance between accuracy and performance. Furthermore, NICMM also 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 that require 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), with broad applications spanning industrial and digital domains. Beyond its pivotal role in CAD/CAM systems for product design and manufacturing, it underpins core techniques in geometric computation and image processing. After decades of dedicated scholarly exploration, the theoretical frameworks and methodological systems for curve design within Euclidean space have matured considerably, evolving into a well-established and sophisticated body of knowledge. Nonetheless, this field remains dynamically evolving, continuing to attract sustained research interest in recent years. A particularly notable trend is the growing focus on curve shape control methods, as exemplified in influential studies such as [1–3]. Concurrently, research into their practical implementation and real-world applications has advanced, as highlighted in works like [4].

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 [3–5]. This research endeavor holds profound theoretical significance and a broad spectrum of practical applications, including mesh segmentation and cutting [6,7], computation of Voronoi diagrams in the manifold [8], shape analysis [9], virtual surgery [10], numerical control (NC) tool path generation [11], and vector graphs [12].

☆ This work was supported by the Key R&D Programs of Zhejiang Province (Nos. 2023C01224).
* Corresponding author at: Zhejiang Sci-Tech University, Hangzhou 310018, China.
** Corresponding authors.
E-mail addresses: 2023220603066@mails.zstu.edu.cn (J. Wang), zhangqi3052@zstu.edu.cn (Q. Zhang), zhangyun@cuz.edu.cn (Y. Zhang), jinyao@zstu.edu.cn (Y. Jin), zhxhz@zstu.edu.cn (H. Zhang), llhe@zju.edu.cn (L. He).

https://doi.org/10.1016/j.cag.2025.104351
Received 27 June 2025; Received in revised form 22 July 2025; Accepted 26 July 2025
Available online 14 August 2025

Despite its potential, designing curves on discrete 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 [13], smoothing [14], parameterization [15], or spline-based approaches [16]. 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) [17] or variational formulations [18] 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 [18]. 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 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. Unlike Physics-Informed Neural Networks (PINNs) [19] that operate on continuous fields, we propose a novel neural implicit method named Neural Implicit Curve Modeling on Meshes (NICMM), specifically designed for discrete geometric learning on surface meshes. 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, leveraging the extensive body of concepts and techniques developed for curve design in 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 preserve fundamental geometric properties but also strictly adhere to the constraints imposed by the manifold surface. Typically, these approaches employ intrinsic or extrinsic optimization strategies to shape the curves, incorporating a diverse range of techniques such as parameterization, smoothing, projection, et al.

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. [15] 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. [7] 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 optimize energy functionals to evolve curves on meshes. Jung et al. [20] first adapted the active contour model of image to meshes, but constrained the curve to the edges. Ji et al. [21] relaxed this constraint through dynamic node operations, improving smoothness. Later approaches employed intrinsic operators: Lawonn et al. [14] used Laplacian smoothing (high computational cost), while Pawellek et al. [5] achieved better convergence via 4D lifting. However, these methods remain challenged by energy functional design under manifold constraints and computational complexity.

Spline-based methods generalize Euclidean splines to discrete manifolds using Riemannian metrics. Early work by Wallner et al. [22] introduced geodesic averaging for subdivision curves, albeit with high computational intensity. Subsequent improvements include: Morera et al. [23] optimized geodesic computations, and Mancinelli et al. [24] developed recursive De Casteljau algorithm for Bézier curves with better continuity. While these approaches advance computational efficiency, they remain limited in shape control and interpolation capability.

Extrinsic projection methods design curves in Euclidean space before projecting onto manifolds. Hofer et al. [25] pioneered this approach using energy-minimizing splines, which was later extended to higher-order energies by Pottmann et al. [26]. Subsequent developments include: Panozzo et al.'s [27] high-dimensional embedding (notably computationally intensive); Jin et al.'s [13] thin-shell space method (robust yet complex); and Xu et al.'s [28] simplified shell space combined with B-splines (featuring improved quality but associated efficiency trade-offs). While these methods offer design flexibility, projection operations may compromise the robustness and quality of the resulting 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 success 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 [29]. Generally, implicit methods are predominantly based on numerical techniques. 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 [30,31], and are often hampered by limited user control.

The implicit curve methods are evolving with notable advances. Wu et al. [17] pioneered an implicit curve evolution method grounded in the geodesic curvature flow equation. To boost convergence, they adopted a semi-implicit integration technique; however, the overall computational efficiency of the method remained subpar. Building on this foundation, Zhang et al. [32] symmetrized the coefficient matrix of the equation and reduced the computational dimension by establishing a narrow-band domain, thereby significantly improving the efficiency of the method and making it more amenable to mesh segmentation tasks. Liu et al. [33] further innovated with a refined discrete approach for geodesic curvature flow. This method sparsified the coefficient matrix and minimized the solution dimension via narrow-band construction, resulting in a significant improvement in efficiency. Most recently, Zhang et al. [18] 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 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 generating high-quality self-intersection-free curves. However, most 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 dependent on the quality of the input mesh, posing a persistent bottleneck for practical applications.

2.3. Neural implicit method

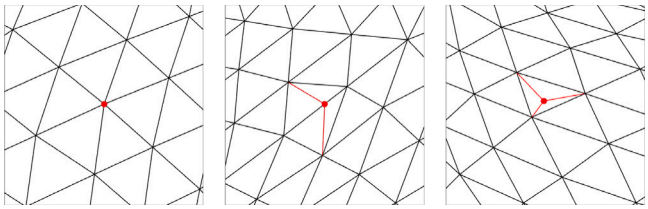
Neural implicit representations have recently gained significant attention in the fields of geometric processing and 3D reconstruction, captivating researchers worldwide. In the recent comprehensive survey, Neural Fields in Visual Computing and Beyond [34], the authors meticulously review coordinate-based neural networks, exploring their applications in representing geometry, appearance, and physical fields across diverse domains. Building on this foundation, [35] demonstrated the potential of multi-layer perceptrons (MLPs) as geometry processors, successfully substituting mesh-based discretization with neural fields for tasks including deformation and filtering.

Within the realm of 3D reconstruction, notable advancements have been made. [36] introduced Occupancy Networks, which model shape boundaries as implicit surfaces, while [37] presented DeepSDF, a continuous signed distance representation for 3D geometry. The seminal work of [38] further propelled this research trajectory forward with the development of NeRF, a neural radiance field model that enables the synthesis of photorealistic views.

Our proposed method adheres to the overarching concept of neural implicit modeling but distinguishes itself through its implementation in discrete mesh domains. In contrast to PINNs [19], which are designed to address continuous PDE-constrained problems, our approach focuses on solving scalar fields on the surface meshes. This fundamental divergence in focus allows our method to effectively leverage the information inherent in meshes, making it particularly well-suited for applications where mesh-based representations are preferred or already available. By integrating neural network architectures with mesh-based operations, we strike a balance among the flexibility of neural implicit models, the computational efficiency of discrete representations, and the geometric fidelity of mesh-based methods.

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:



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

Fig. 1. Illustration of control points in different positions.

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. [18] proposed an effective 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, named NICMM. 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) input features construction; (2) network architecture design; and (3) loss functions design.

3.1. Input feature construction

While neural implicit methods typically take point coordinates as network input, often in combination with latent codes or positional encodings. We further enhance the input representation by explicitly encoding constraint information and incorporating geometric features derived from the mesh structure. This enriched input facilitates more effective feature learning and accelerates convergence.

The process begins with the construction of a sparse interpolation scalar field ϕ_p, which is based on the control point sequence $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{k}$. 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.

After performing the aforementioned operations, the original mesh $\mathcal{M} = (V, F)$ is transformed into an updated mesh $\mathcal{M}' = (V', F')$, 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:

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

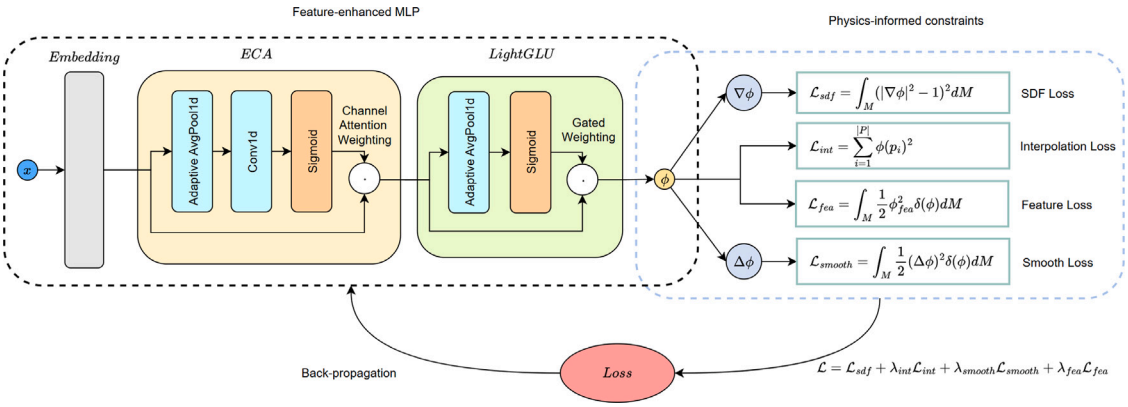


Fig. 2. NICMM architecture diagram.

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 improve efficiency and accelerate convergence, an initial Signed Distance Field (SDF) is computed and used as the basis for network initialization. The construction of this initial SDF involves the following steps: First, a closed-loop path (represented as a polygonal curve) is constructed to traverse the control points along the mesh edges associated with $\mathcal{P}$. For each adjacent point pair, the shortest path is calculated using Dijkstra's algorithm, and these paths are concatenated to form a polygonal loop on the mesh vertices. It should be noted that this procedure may generate self-intersections or nested loops due to the complexity of the underlying mesh topology and geometry. In such cases, we explicitly check the validity of the loop. If the loop is simple (i.e., non-self-intersecting and non-nested), a flood-fill algorithm is applied to determine the indices of the vertices enclosed by the loop. We then compute an unsigned distance field (UDF) using the heat method [39], with the closed-loop vertices treated as the source set. Finally, the SDF ϕ_0 is obtained by assigning negative values to the UDF entries corresponding to vertices inside the loop. If the loop is invalid (i.e., self-intersecting or nested), the sign assignment step is skipped, and the UDF is directly used as the initial scalar field input to the network.

The initial feature vector $x = [\mathbf{P}, \mathbf{N}, \phi_p, \phi_0]$ is formed by concatenating the vertex coordinates $\mathbf{P}$ and the vertex normals $\mathbf{N}$ of the input mesh, together with the scalar fields ϕ_p and ϕ_0. Serving as network input, this feature vector encapsulates the geometric and constraint-related information required for subsequent processing.

3.2. Network architecture design

We propose the Neural Implicit Curve Modeling on Meshes (NICMM) network, a novel architecture designed for geometric deep learning on surface meshes in three-dimensional space. Unlike conventional neural approaches that rely on generic multilayer perceptrons (MLPs), NICMM incorporates a hierarchical feature processing architecture Fig. 2 to better capture geometric features and accelerate convergence. Specifically, the architecture enhances the basic MLP structure with the following specialized modules:

- Efficient Channel Attention (ECA): To improve the representation capability of the network without introducing significant computational overhead, we incorporate the Efficient Channel Attention (ECA) mechanism after each convolutional block. Specifically, for a feature map with C channels, ECA first applies global average pooling along the spatial (or sequence) dimension to obtain a channel-wise descriptor. Then, a lightweight 1D convolution with an adaptively determined kernel size k is applied to capture local cross-channel interactions:

$$k = |\frac{log_2 C + b}{\gamma}|, \tag{2}$$

where γ and b are hyperparameters controlling the kernel size. The result is passed through a sigmoid function to obtain attention weights, which are then applied to the original features via channel-wise multiplication. This operation enhances informative channels while suppressing less relevant ones, improving the discriminative ability of the learned representation. The architecture of this module is shown in the yellow-colored component of Fig. 2.

- Light Gate Linear Unit (Light GLU): To further enhance feature selectivity after channel attention, we adopt a simplified gated activation mechanism, LightGLU. It applies a learnable linear transformation followed by a sigmoid function to generate channel-wise gating weights, which are used to modulate the input features via element-wise multiplication. This module complements ECA by providing additional non-linear refinement, enabling more precise control over the feature representation with minimal overhead. Its implementation is visualized in the green-colored component of Fig. 2.

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 an implicit curve involves the combined differential operators, such as gradients and Laplacian of the scalar field. Although traditional methods, such as PINNs utilizing automatic differentiation and tangent plane projection, can be used for discretization, they may introduce significant errors that reduce the accuracy of the solution. To address this, we design a total loss function that integrates multiple geometric constraints inspired by variational principles. Our formulation mainly draws on the energy proposed by Zhang et al. [18], and employs a finite element-based discretization to fully exploit the underlying mesh structure. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sdf}} + \lambda_{\text{int}}\mathcal{L}_{\text{int}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}} + \lambda_{\text{fea}}\mathcal{L}_{\text{fea}}. \tag{3}$$

Each term in this loss function captures a specific geometric prior or user constraint:

The first term is derived from the Eikonal equation and aims to regularize the predicted signed distance field (SDF) by enforcing a unit gradient magnitude. Following [18], we adopt a numerically stable smooth energy form:

$$\mathcal{L}_{\text{sdf}} = \int_M (|\nabla\phi|^2 - 1)^2 dM. \tag{4}$$

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

To strictly enforce user-specified control points, we penalize the deviation of ϕ at those points:

$$\mathcal{L}_{\text{int}} = \sum_{i=1}^{|P|} \phi(p_i)^2. \tag{5}$$

We encourage the smoothness of the resulting curve by minimizing its geodesic curvature energy. Specifically, we use the Willmore energy formulation:

$$\mathcal{L}_{\text{smooth}} = \int_{\Gamma} \kappa_g^2 dl = \int_{M} \kappa_g^2(\phi)\delta(\phi)|\nabla\phi| d\boldsymbol{M}, \tag{6}$$

where κ_g represents the geodesic curvature, and $\delta(\phi)$ 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 $|\nabla\phi| = 1$ can be treated as a prior constraint. This simplifies the above formula and approximates the geodesic curvature κ_g using the Laplacian operator of the level set function ϕ leading to the following simplified form:

$$\mathcal{L}_{\text{smooth}} = \int_{\Gamma} \kappa_g^2 dl = \int_{M} \frac{1}{2}(\Delta\phi)^2\delta(\phi) d\boldsymbol{M}. \tag{7}$$

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 [28], to extract salient structural features such as sharp edges or ridges.

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 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.

$$\mathcal{L}_{\text{fea}} = \int_{M} \phi_{fea}\delta(\phi) d\boldsymbol{M}. \tag{8}$$

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.

Traditional neural field methods typically require sampling points within a continuous domain, posing challenges for mesh-based applications. To address this, we employ a discretization strategy where all mesh vertices serve as computational nodes. Unlike continuous domain approaches, our method computes loss functions over the Voronoi domains associated to these nodes, enabling robust numerical integration and improved optimization stability. This discretization scheme is applied to the level set loss, smoothness loss, and feature-aware loss terms as follows:

$$\begin{aligned}
E_{\text{sdf}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \frac{1}{2}(|\nabla\phi(c_t)|^2 - 1)^2, \\
E_{\text{smooth}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \frac{1}{2}(\Delta\phi(c_t))^2 \cdot G_\sigma(\phi(c_t)), \\
E_{\text{fea}}(\phi) &\approx \sum_{i=1}^{|V|} \sum_{t\in\mathcal{N}(v)} w_{i,t} \frac{1}{2}\phi_{fea} \cdot G_\sigma(\phi(c_t)),
\end{aligned} \tag{9}$$

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 to $|t|/3$, $G_\sigma(\phi(c_t))$ 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 RTX 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 algorithm 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 [40]. 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:

$$\Theta^* = arg\min_{\Theta} \sum_{i}^{N} \mathcal{L}_{\text{total}}(\phi_\Theta(\mathbf{P}_i)), \tag{10}$$

where, $\phi_\Theta(\mathbf{P}_i)$ is the predicted result of the network at the control point $\mathbf{P}_i$, 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 mesh under evaluation, we randomly sample 1000 sets of control points from its vertex set, each set comprising

Table 1
Comparison of average loss for varying prediction durations. Bold indicates the optimal value.

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**

Table 2
Comparison of network with and without pretraining.

Pretraining duration	Average loss
Without Pretraining	15.298
Pretrained for 30 min	10.996

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 min. 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:

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 connectivity 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 network 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.

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 meets the 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 stopping 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 early stopping strategyillustrated in Fig. 3. The loss function notably plateaus after 200 iterations.

To evaluate the impact of the early stopping 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 min of pre-training time, and 70 rapid convergence iterations. The results are tabulated in Table 3:

Table 3
Convergence efficiency across model variants. Bold indicates optimal values.

Network	Time (ms)	Loss
NICMM	109	**8.29**
w/o ECA	98	13.11
w/o LightGLU	93	12.26
w/o Both	**87**	16.32

To ensure fair comparison across model variants, we conducted time-aligned ablation studies by normalizing the training duration. For each ablated model, we first calculated its per-iteration time cost using the original experimental data. This allowed us to determine the equivalent number of iterations achievable within the complete model's training timeframe (70 iterations at 109 ms). The adjusted iteration counts were rounded to nearest integer for practical implementation. The relative performance difference was quantified as:

$$\Delta\text{Loss} = \frac{\text{Loss}_{\text{variant}} - \text{Loss}_{\text{NICMM}}}{\text{Loss}_{\text{NICMM}}} \times 100\%. \tag{11}$$

As shown in Table 4, when trained for equal durations, the full NICMM architecture retains its performance advantage. Ablated models

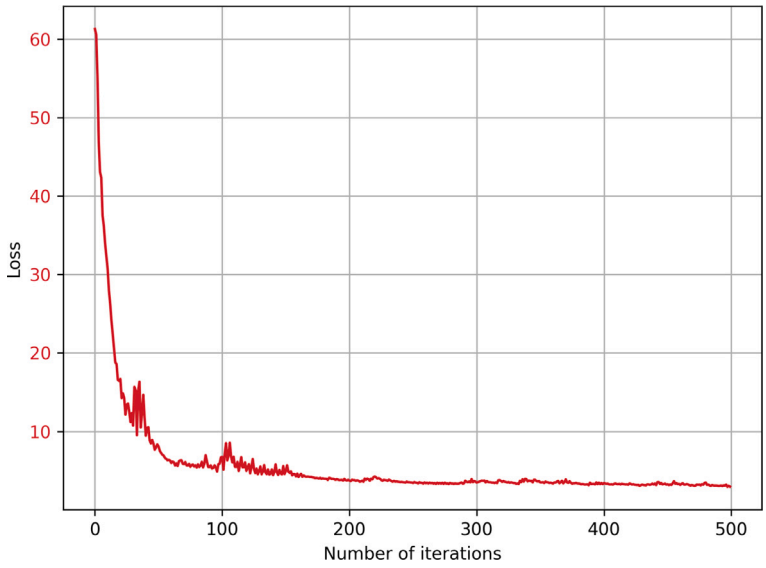


Fig. 3. Change of loss with respect to the number of iterations.

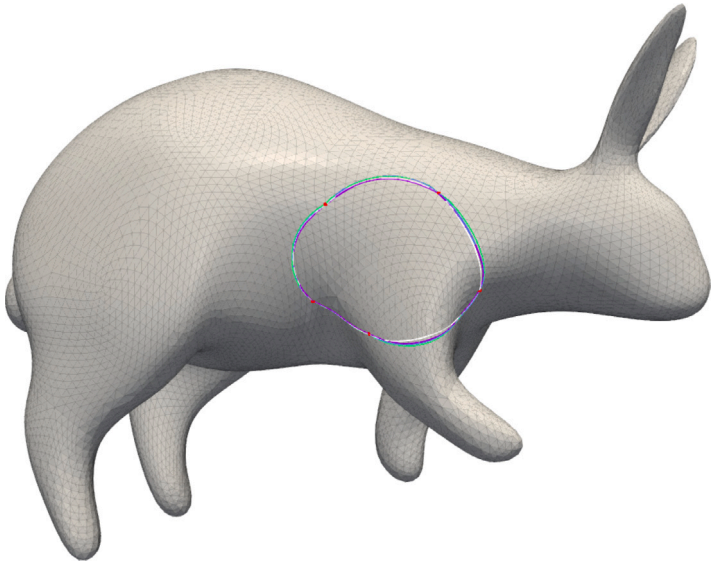


Fig. 4. The figure shows four performance curves with different prediction frequencies: blue (41 iterations, meeting early-stopping criteria), orange (100 iterations), green (200 iterations), and purple (500 iterations).

Table 4
Time-aligned comparison of model variants. Models trained until reaching NICMM's training time. Bold indicates optimal values.

Network	Iterations	Loss	Δ Loss vs NICMM
NICMM	70	**8.29**	–
w/o ECA	78	12.30	+48.3%
w/o LightGLU	82	10.34	+24.7%
w/o Both	88	14.51	+75.0%

Table 5
Comparison of prediction with different initial features. Bold indicates the best performance.

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

exhibit significantly higher loss values despite their increased iteration counts. The w/o both variant shows the largest performance degradation. This confirms that the NICMM's superior performance stems from its architectural completeness rather than extended training time. The persistent performance gaps under time-aligned conditions demonstrate each module's substantive contribution to model capability.

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: 40 test samples and 2 min of pre-training. The results are presented in Table 5.

Table 6
Comparison of final prediction losses using different initial features after 50 training iterations. Bold indicates the best performance.

Initial feature combination	Average final loss
Full Features	**9.10**
w/o Vertex Normals	9.38
w/o Interpolation Scalar Field	11.84
w/o Initial Level Set	28.41

As shown in Table 5, the network achieves the early stopping 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.

To further assess the impact of different input features on convergence behaviors, we conducted an additional ablation experiment that evaluates the average prediction losses after a fixed number of training iterations. Specifically, each model variant was trained for a consistent number of 50 iterations using identical settings, and the final loss was recorded over 40 test samples. This provides a direct measure of how different initial feature combinations affect the quality of convergence under the same optimization budget. The results, summarized in Table 6, demonstrate that complete input features lead to the lowest average prediction loss, corroborating the earlier iteration-based findings. In particular, omitting either the interpolation scalar field or the initial level set results in significantly higher final losses, suggesting that these features play a critical role in guiding the optimization process toward better minima.

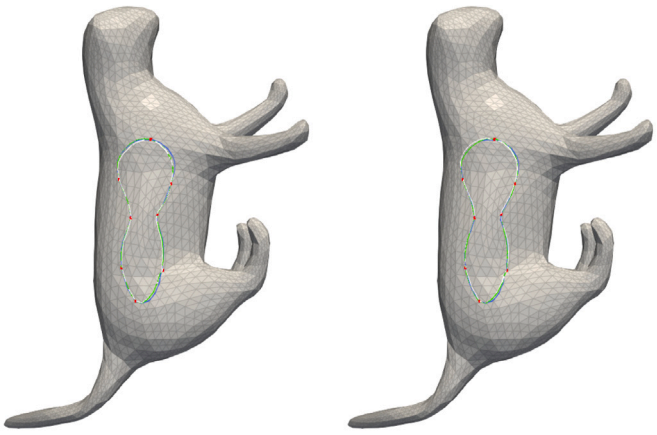
4.3. Parameters

In the loss function (Eq. (3)), 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.

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, which is 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.



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

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

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.

4.5. Comparisons

Given that NICMM's physical constraints are consistent with those of traditional optimization-based methods [18], 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 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 [18] and subsequently discretized into polyline representations. Using these representations, the geodesic curvature κ_g is calculated at each vertex.

$$\kappa_g = \frac{2sin\theta_i}{|\hat{r}_{i+1} - \hat{r}_{i-1}|}, \tag{12}$$

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 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

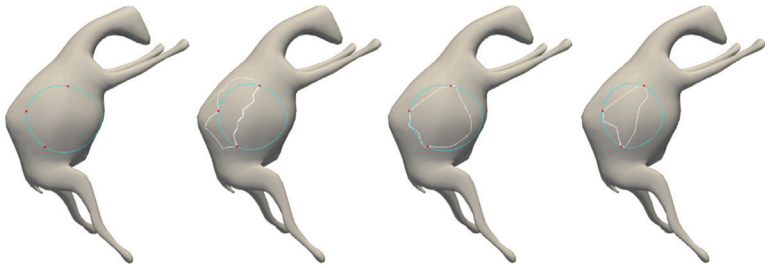


Fig. 6. The white curves in the figure represent manually constructed initial closed loops, while the blue curves denote the predicted results by the network. The leftmost image shows a case without an initial loop, using only interpolation points to generate the signed distance field.

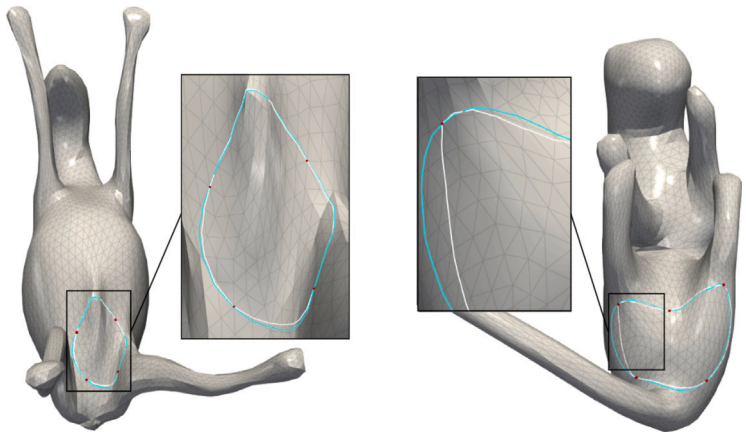


Fig. 7. Comparison of variational methods.

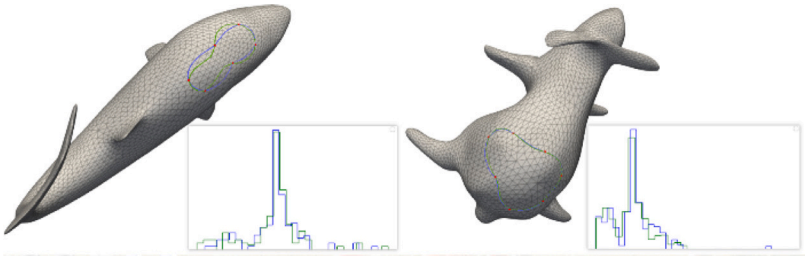


Fig. 8. Comparison of geodesic curvature across different methods.

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.

To further evaluate the smoothness and geometric consistency of generated curves, we adopt an alternative geodesic curvature metric κ_{g_a} that is based on angle measurement [41] rather than pointwise differential geometry. This metric is defined as:

$$\kappa_{g_a} = \begin{cases} \pi - \dfrac{2\pi\beta}{\theta}, & \text{if point on vertex,} \\ \pi - \beta, & \text{if point on edge,} \end{cases} \tag{13}$$

where θ denotes the total angle around the vertex (i.e., the sum of corner angles of adjacent triangles), and β is the smaller of the two interior angles formed by the curve at that point.

Based on this curvature metric, we define the average curve quality using the following integral form:

$$\overline{\kappa}_{g_a} = \frac{\int \kappa_{g_a}\, ds}{L} = \sum_{i=1}^{N} \frac{|\kappa_{g_a}(v_i)| \cdot l(v_i)}{L}, \tag{14}$$

Table 7
Comparison of runtime (averaged over three runs) between NICMM and the variational method (VM) on the test data.

Method	Test Set 1	Test Set 2
NICMM	**6.89**	**8.01**
VM	8.34	15.54

where, v_i denotes the ith vertex of the discretized curve, defined as the ordered sequence of intersections between the zero level-set and mesh edges, $\kappa_{g_a}(v_i)$ is its corresponding angle-based geodesic curvature, $l(v_i)$ is the arc length associated with v_i (half the sum of adjacent edge lengths), and $L = \sum_{i=1}^{N} l(v_i)$ represents the total length of the curve. This formulation provides a compact, geometry-aware evaluation of average curve smoothness and surface adherence.

Controlled experiments were conducted on two test sets derived from 50 base models. Test Set 1 comprises meshes of relatively high quality, while Test Set 2 consists of meshes generated by applying perturbations to those in Test Set 1 using the method outlined in Section 4.6. The experimental results are presented in Table 7.

Table 8
Comparison of runtime (averaged over three runs) between NICMM and the variational method (VM) on the test data.

Method	Avg (ms)	Std (ms)	Min (ms)	Max (ms)
NICMM	105	**22**	96	**245**
VM	**102**	106	**24**	538

As illustrated in Table 7, NICMM exhibits comparable or superior performance relative to VM in both test sets. For meshes of Test Set 1, NICMM yields a marginally lower average value (6.89 versus 8.34). More strikingly, on meshes from Test Set 2, NICMM outperforms VM significantly, with an average improvement of 94.0% (8.01 versus 15.54). Additionally, VM failed to generate closed curves in 6 out of 50 degraded cases, whereas NICMM achieved success in all 50 cases. This highlights its tighter curvature control and robustness against severe geometric artifacts. These results further support the assumption that the quality of outcomes produced by VM is highly sensitive to mesh quality, whereas our approach remains unaffected.

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 8.

Although NICMM requires backpropagation through the entire network during optimization, recent improvements such as CUDA Graph integration and low-level code optimizations have significantly enhanced its efficiency. In particular, our fine-tuning strategy, where the same input is repeatedly optimized, allows CUDA Graph to effectively cache and reuse computational graphs. This leads to substantial speedups despite multiple backward passes. As shown in our experiments, the average runtime (105 ms) is now comparable to that of the traditional variational method (102 ms). Moreover, NICMM exhibits a significantly lower standard deviation (22 ms vs. 106 ms) and smaller worst-case latency (245 ms vs. 538 ms), making it not only efficient but also more stable in practice.

To further validate the advantages of the proposed NICMM framework in curve modeling, we perform a comparative analysis against B-surf [16], 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 requires 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.

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 triangles. 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, restricted 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 meshes using traditional methods with numerical optimization (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 mitigates 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 faced with nearly degenerate mesh elements. 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.

To assess the stability of NICMM on surfaces 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

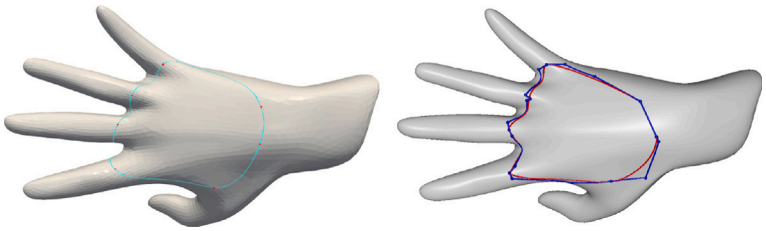


Fig. 9. Comparison with explicit methods. Left: Smooth implicit curve generated by NICMM from user-specified control points, exhibiting geometric fairness with uniform curvature distribution. Right: Explicit spline curve produced by B-surf under identical input points, with overlaid Bézier control points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

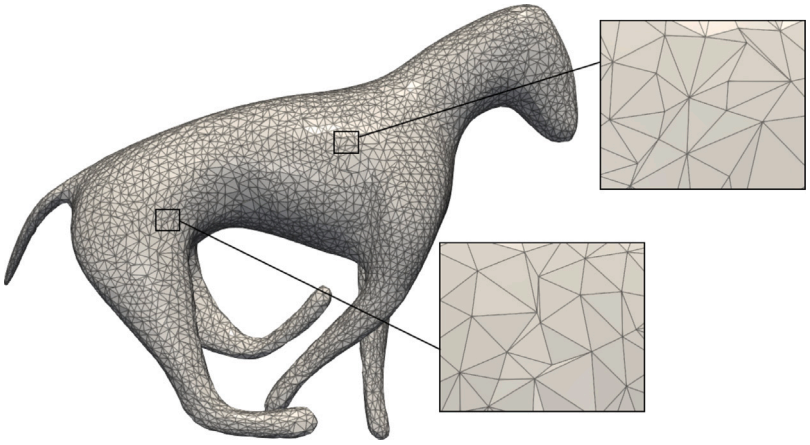


Fig. 10. Schematic diagram of low-quality mesh. The black bounding boxes highlight regions containing nearly degenerate patches.

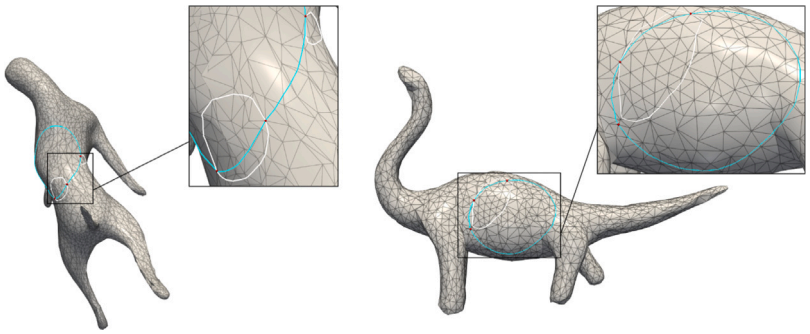


Fig. 11. Comparison results on low-quality meshes. NICMM (blue) vs. traditional variational method (white).

in the upper row of Fig. 12, where all curves successfully interpolate control points while maintaining nearly identical shapes, even when the mesh 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 the 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 field, the curve successfully avoids undesired areas, demonstrating the unified treatment of different types of guidance.

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

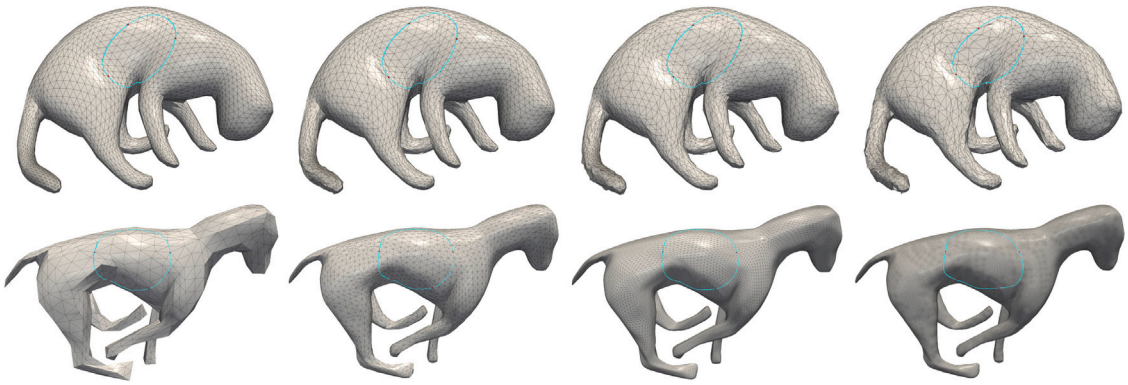


Fig. 12. Illustration of the robustness of NICMM under varying mesh qualities and resolutions. The top row shows mesh surfaces with different levels of geometric perturbations applied to a high-quality base mesh. The bottom row presents the corresponding curves predicted by NICMM on meshes with varying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

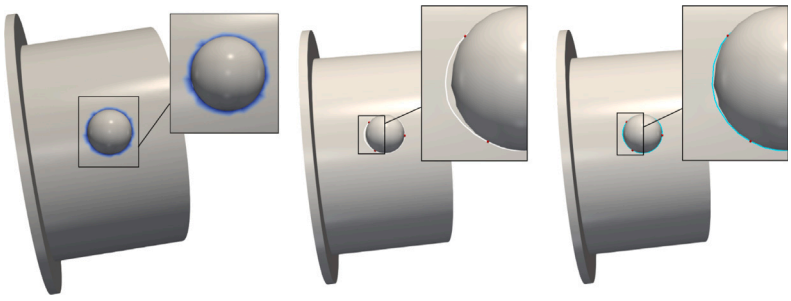


Fig. 13. From left to right: input with alignment region (blue), result without guidance, and result with feature alignment.

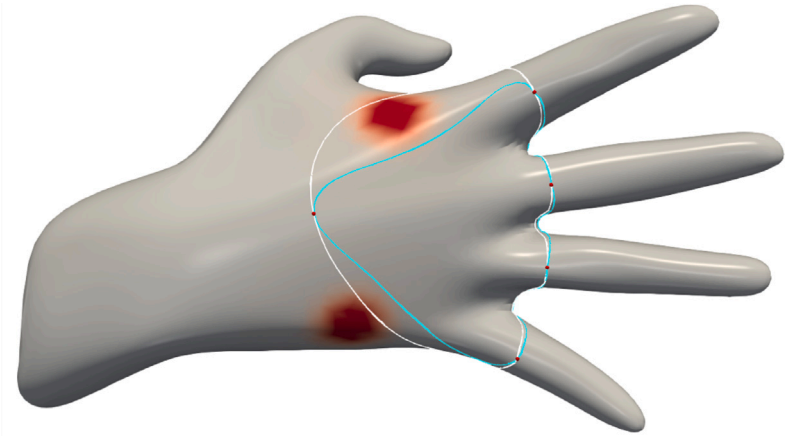


Fig. 14. In the figure, the white and blue curves represent the network's predictions for the same control points without and with obstacle constraints.

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. Application

Our curve modeling algorithm exhibits broad applicability across diverse domains. As a representative illustration, we demonstrate its efficacy in refining mesh segmentation results.

Existing mesh segmentation approaches, such as those presented in [42,43], predominantly assign class labels to individual mesh faces, partitioning the entire mesh into distinct regions. Nevertheless, these methods inherently restrict region boundaries to mesh edges. Owing to the discrete nature of meshes, this often culminates in jagged or irregular boundary formations, as vividly depicted in Fig. 16. Although techniques like geodesic loop extraction [44] can yield smoother boundaries in certain scenarios, they remain highly sensitive to local curvature variations. Consequently, these methods frequently struggle to accurately delineate convex transitions and complex topological features.

To empirically validate the practical utility of our smooth curve generation methodology, we conducted a boundary smoothing experiment on the COSEG dataset. This dataset encompasses mesh segmentation models with meticulously annotated ground-truth face-level region information. For each mesh within the dataset, we first extracted region boundaries by pinpointing the edges that separate faces of different labels. Subsequently, we collated ordered point sequences along these boundary edges, which served as dense interpolation points

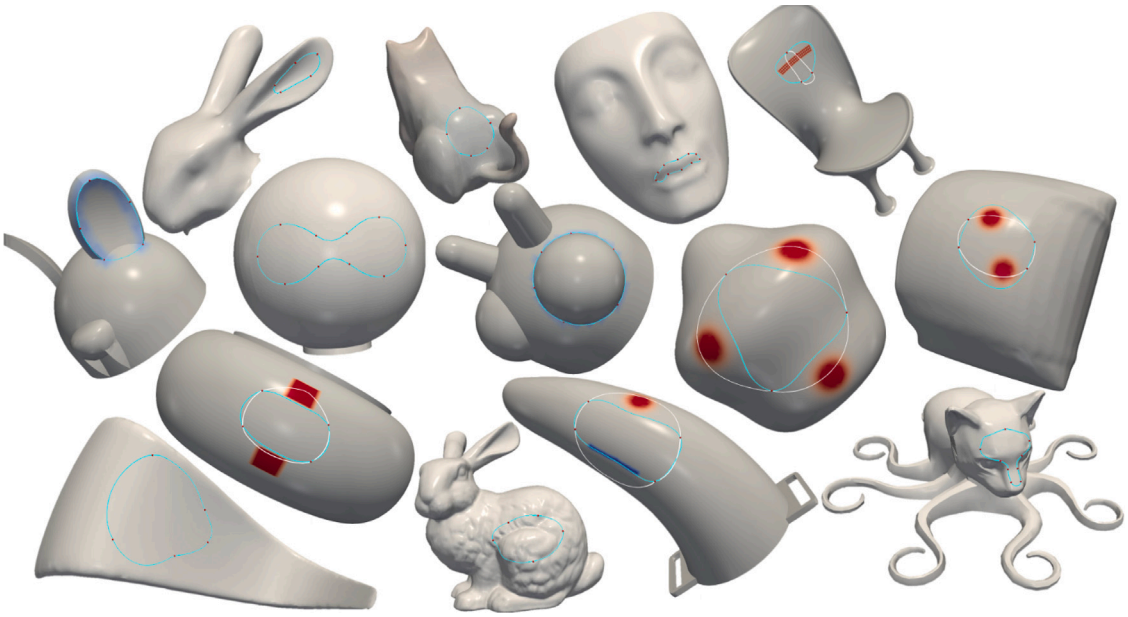


Fig. 15. The gallery of curve design results by NICMM on various meshes. The red regions marked on the models indicate obstacle areas, and the blue regions indicate feature alignment areas. The white curves are generated without feature constraints.

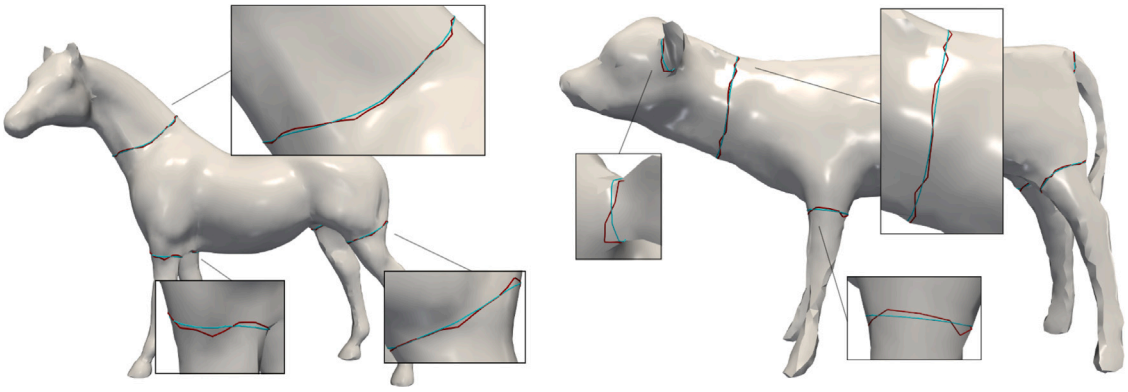


Fig. 16. Boundary smoothing results on two COSEG segmentation models. The red curve indicates the original segmentation boundary along mesh edges, while the blue curve represents the smooth boundary predicted by our method (NICMM) using dense edge points with relaxed interpolation.

for our model. The resultant boundaries, typically composed of numerous short, contiguous edge segments, pose significant challenges to traditional interpolation techniques.

To address the high density and close proximity of these interpolation points, we strategically relaxed the interpolation constraint by setting a relatively low interpolation weight $\lambda_{\text{int}} = 500$. This adjustment enabled our model to approximate rather than strictly interpolate the points, thus generating boundaries that are not only smoother and more visually consistent but also transcend the limitations of the original mesh edges, as clearly shown in Fig. 16. This experiment not only exemplifies a powerful downstream application of our method, transforming coarse, discrete segmentation boundaries into smooth curves while preserving the original region assignments, but also showcases its unique advantage. By enabling more accurate and aesthetically pleasing segmentation outcomes without altering the fundamental partitioning, our approach can significantly enhance downstream tasks, including mesh editing, visualization, and physical simulation. Additionally, the experiment underscores the robustness of our model in dealing with densely distributed and irregular interpolation constraints, thereby highlighting its potential for seamless integration into existing mesh segmentation workflows.

6. Conclusion

We propose NICMM, which integrates deep learning techniques with traditional implicit curve design methodologies. This innovative network takes triangular meshes and control points as inputs, computing a level set function that adheres to multiple geometric constraints. Consequently, it enables the generation of curves that lie precisely on surface meshes. To optimize the performance of the network, we have devised a two-stage training strategy. In the initial stage, a concise pretraining process is conducted 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 results incontrovertibly validate that NICMM excels significantly in both robustness and generalization performance while maintaining competitive efficiency. Notably, it maintains consistent performance across meshes with disparate discretization schemes, highlighting its adaptability to various geometric representations. Even when applied to low-quality meshes marred by elongated faces or acutely small angles, NICMM adeptly generates smooth and geometrically valid curves, underscoring its exceptional resilience against mesh degradation. Moreover, the model exhibits remarkable insensitivity to the initial level set construction. Irrespective of the initialization

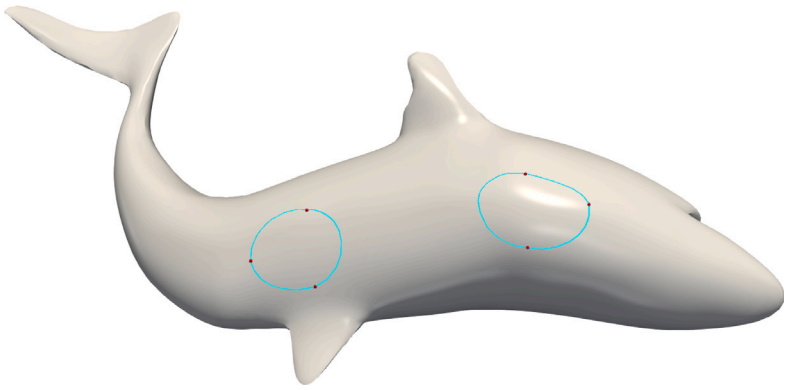


Fig. 17. Predicted curve contains multiple loops due to the lack of topological control.

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

Despite these advances, the proposed approach has limitations and several critical challenges remain ripe for further investigation. Firstly, while the current network exhibits promising capabilities, its efficiency has not yet shown significant improvement compared to traditional numerical methods. To address this, future research will focus on a data-driven framework and exploring more efficient and streamlined network architectures, aiming to facilitate swift and direct curve prediction. Secondly, the network's inability to exert explicit control over the topologies of generated curves poses a notable drawback. As depicted in Fig. 17, this limitation may result in unanticipated topology, 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. Doing so 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.

CRediT authorship contribution statement

Jintong Wang: Writing – review & editing, Writing – original draft, Software. **Qi Zhang:** Formal analysis. **Yun Zhang:** Investigation. **Yao Jin:** Methodology, Investigation, Funding acquisition. **Huaxiong Zhang:** Resources. **Lili He:** Supervision, Software.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yao Jin reports article publishing charges was provided by Zhejiang Sci-tech University. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

Data availability

Data will be made available on request.

References

[1] Yuksel Cem. A class of c^2 interpolating splines. ACM Trans Graph 2020;39(5):1–14.

[2] Zhu Haikuan, Cao Juan, Xiao Yanyang, Chen Zhonggui, Zhong Zichun, Zhang Yongjie Jessica. TCB-spline-based image vectorization. ACM Trans Graph 2022;41(3):1–17.

[3] Mancinelli Claudio, Puppo Enrico. Splines on manifolds: A survey. Comput Aided Geom Design 2024;112:102349.

[4] Marin D, Maggioli F, Melzi S, Ohrhallinger S, Wimmer M. Reconstructing curves from sparse samples on Riemannian manifolds. Comput Graph Forum 2024;43(5):e15136.

[5] Pawellek M, Rössl C, Lawonn K. Distance-based smoothing of curves on surface meshes. Comput Graph Forum 2024;43(5):e15135.

[6] Kaplansky Lotan, Tal Ayellet. Mesh segmentation refinement. Comput Graph Forum 2009;28(7):1995–2003.

[7] Lee Yunjin, Lee Seungyong, Shamir Ariel, Cohen-Or Daniel, Seidel Hans-Peter. Mesh scissoring with minima rule and part salience. Comput Aided Geom Design 2005;22(5):444–65.

[8] Xin Shiqing, Wang Pengfei, Xu Rui, Yan Dongming, Chen Shuangmin, Wang Wenping, et al. SurfaceVoronoi: Efficiently computing voronoi diagrams over mesh surfaces with arbitrary distance solvers. ACM Trans Graph 2022;41(6):1–12.

[9] Gehre A, Bronstein M, Kobbelt L, Solomon J. Interactive curve constrained functional maps. Comput Graph Forum 2018;37(5):1–12.

[10] Zachow Stefan, Gladilin Evgeny, Sader Robert, Zeilhofer Hans-Florian. Draw and cut: intuitive 3D osteotomy planning on polygonal bone models. Int Congr Ser 2003;1256:362–9.

[11] Lasemi Ali, Xue Deyi, Gu Peihua. Recent development in CNC machining of freeform surfaces: A state-of-the-art review. Computer- Aided Des 2010;42(7):641–54.

[12] Mancinelli Claudio, Puppo Enrico. Vector graphics on surfaces using straightedge and compass constructions. Comput Graph 2022;105:46–56.

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

[14] Lawonn Kai, Gasteiger Rocco, Rössl Christian, Preim Bernhard. Adaptive and robust curve smoothing on surface meshes. Comput Graph 2014;40:22–35.

[15] Lee Yunjin, Lee Seungyong. Geometric Snakes for Triangular Meshes.

[16] Mancinelli Claudio, Nazzaro Giacomo, Pellacini Fabio, Puppo Enrico. B/surf: Interactive Bézier splines on surface meshes. IEEE Trans Vis Comput Graphics 2023;29(7):3419–35.

[17] Chunlin Wu, Xuecheng Tai. A level set formulation of geodesic curvature flow on simplicial surfaces. IEEE Trans Vis Comput Graphics 2010;16(4):647–62.

[18] Zhang Xiaohu, Wu Shuang, Chen Jiong, Jin Yao, Bao Hujun, Huang Jin. Versatile curve design by level set with quadratic convergence. IEEE Trans Vis Comput Graphics 2024;1–10.

[19] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[20] Jung Moonryul, Kim Haengkang. Snaking across 3d meshes. In: 12th Pacific conference on computer graphics and applications, 2004. PG 2004. proceedings. IEEE; 2004, p. 87–93.

[21] Ji Zhongping, Liu Ligang, Chen Zhonggui, Wang Guojin. Easy mesh cutting. In: Computer graphics forum, vol. 25, (3):Wiley Online Library; 2006, p. 283–91.

[22] Wallner Johannes, Pottmann Helmut. Intrinsic subdivision with smooth limits for graphics and animation. ACM Trans Graph 2006;25(2):356–74.

[23] Morera Dimas Martínez, Carvalho Paulo Cezar, Velho Luiz. Modeling on triangulations with geodesic curves. Vis Comput 2008;24(12):1025–37.

[24] Mancinelli Claudio, Puppo Enrico. Computing the Riemannian center of mass on meshes. Comput Aided Geom Design 2023;103:102203.

[25] Hofer Michael, Pottmann Helmut. Energy-Minimizing Splines in Manifolds.

[26] Pottmann Helmut, Hofer Michael. A variational approach to spline curves on surfaces. Comput Aided Geom Design 2005;22(7):693–709.

[27] Panozzo Daniele, Baran Ilya, Diamanti Olga, Sorkine-Hornung Olga. Weighted averages on surfaces. ACM Trans Graph 2013;32(4):1–12.

[28] Xu Rongyan, Jin Yao, Zhang Huaxiong, Zhang Yun, Lai Yu-kun, Zhu Zhe, et al. A variational approach for feature-aware B-spline curve design on surface meshes. Vis Comput 2023;39(8):3767–81.
[29] Gibou Frederic, Fedkiw Ronald, Osher Stanley. A review of level set methods and some recent applications. J Comput Phys 2018;353:82–109.
[30] Hu Ping, Shuai Bing, Liu Jun, Wang Gang. Deep level sets for salient object detection. In: 2017 IEEE conference on computer vision and pattern recognition. Honolulu, HI: fffffflEEE; 2017, p. 540–9.
[31] Wang Zian, Acuna David, Ling Huan, Kar Amlan, Fidler Sanja. Object instance annotation with deep extreme level set evolution. In: 2019 IEEE/CVF conference on computer vision and pattern recognition. Long Beach, CA, USA: IEEE; 2019, p. 7492–500.
[32] Zhang Juyong, Wu Chunlin, Cai Jianfei, Zheng Jianmin, Tai Xue-cheng. Mesh snapping: Robust interactive mesh cutting using fast geodesic curvature flow. Comput Graph Forum 2010;29(2):517–26.
[33] Liu Zheng, Zhang Huayan, Wu Chunlin. On geodesic curvature flow with level set formulation over triangulated surfaces. J Sci Comput 2017;70(2):631–61.
[34] Xie Yiheng, Takikawa Towaki, Saito Shunsuke, Litany Or, Yan Shiqin, Khan Numair, et al. Neural fields in visual computing and beyond. In: Computer graphics forum, vol. 41, (2):Wiley Online Library; 2022, p. 641–76.
[35] Yang Guandao, Belongie Serge, Hariharan Bharath, Koltun Vladlen. Geometry processing with neural fields. Adv Neural Inf Process Syst 2021;34:22483–97.
[36] Mescheder Lars, Oechsle Michael, Niemeyer Michael, Nowozin Sebastian, Geiger Andreas. Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 4460–70.
[37] Park Jeong Joon, Florence Peter, Straub Julian, Newcombe Richard, Lovegrove Steven. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 165–74.
[38] Mildenhall Ben, Srinivasan Pratul P, Tancik Matthew, Barron Jonathan T, Ramamoorthi Ravi, Ng Ren. Nerf: Representing scenes as neural radiance fields for view synthesis. Commun ACM 2021;65(1):99–106.
[39] Crane Keenan, Weischedel Clarisse, Wardetzky Max. The heat method for distance computation. Commun ACM 2017;60(11):90–9.
[40] Hanocka Rana, Hertz Amir, Fish Noa, Giryes Raja, Fleishman Shachar, Cohen-Or Daniel. Meshcnn: A network with an edge. ACM Trans Graph 2019;38(4):1–12, arXiv:1809.05910 [cs].
[41] Lawonn Kai, Gasteiger Rocco, Rössl Christian, Preim Bernhard. Adaptive and robust curve smoothing on surface meshes. Comput Graph 2014;40:22–35.
[42] Chen Xiaobai, Golovinskiy Aleksey, Funkhouser Thomas. A benchmark for 3D mesh segmentation. ACM Trans Graph 2009;28(3):1–12.
[43] Mu Anyu, Liu Zhenyu, Duan Guifang, Tan Jianrong. Part-to-surface mesh segmentation for mechanical models based on multi-stage clustering. Computer-Aided Des 2023;162:103545.
[44] Xin Shi-Qing, He Ying, Fu Chi-Wing. Efficiently computing exact geodesic loops within finite steps. IEEE Trans Vis Comput Graphics 2011;18(6):879–89.