Engineering with Computers
https://doi.org/10.1007/s00366-025-02199-9

ORIGINAL ARTICLE

DRL-MeshGen: automated block-structured mesh generation framework via deep reinforcement learning and optimal conformal mapping

Long Qi[1,2] · Qiang Wu[2] · Gang Xu[1] · Rushuang Mu[2] · Yang Liu[2] · Jingying Qiu[2] · Jiamin Xu[1] · Renshu Gu[1] · Yufei Pang[2]

Received: 12 June 2025 / Accepted: 17 August 2025

Abstract
Block-structured mesh generation offers significant advantages in numerical computation and simulation, yet conventional methods often rely on manual intervention, struggling to balance automation with high-quality output. To address this, an automated block-structured mesh generation framework that integrates deep reinforcement learning and optimal conformal mapping techniques is proposed in this paper. This framework utilizes the triangulation of the geometric model as input and operates in the following four steps. Firstly, surface triangular meshes are mapped to planar parametric domains using the Ricci flow algorithm. Secondly, isocontours are extracted based on density variation before and after conformal mapping to guide partitioning. Thirdly, a reinforcement learning decision framework is constructed to formulate mesh generation as a sequential decision-making process, where topological template selection and singularity placement are optimized through reward functions. Finally, surface-structured meshes are generated through mesh smoothing and inverse conformal mapping. Experimental results demonstrate that the proposed method outperforms existing methods in both generation efficiency and mesh quality, providing an intelligent new solution for automated mesh generation in CAD/CAE applications.

Keywords Discrete Ricci flow · Contour extraction · Deep reinforcement learning · Topological partitioning · Block-structured mesh

1 Introduction

Mesh generation is essential for carrying out numerical computations and visualizing the results of numerical simulations of computational fluid dynamics (CFD). The accuracy and dependability of CFD simulation results are directly influenced by the quality of the meshes that are produced [1]. Mesh scales have increased dramatically in tandem with the complexity of engineering issues and the ongoing improvement of simulations, resulting in a sharp increase in the labor costs associated with conventional mesh generation techniques. This is now a major obstacle limiting the effectiveness of numerical simulations of CFD. According to statistics, the labor time cost of the meshing step frequently accounts for up to 80% of the total analysis process in complicated industrial design fields like shipbuilding, automotive, and aerospace [2]. Therefore, to meet the growing demand for high-quality meshes, achieving automated, intelligent, and efficient mesh generation has become a research hotspot and urgent requirement of common concern in both academic and industrial circles.

Traditional block-structured mesh generation techniques can be primarily classified into three categories: algebraic mapping algorithms, domain decomposition strategies, and partial differential equation (PDE) approaches. The algebraic mapping approach [3, 4] constructs grids by developing explicit or implicit transformation functions from the

✉ Gang Xu
gxu@hdu.edu.cn

✉ Yufei Pang
grideyes@foxmail.com

[1] School of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, Zhejiang, China

[2] Institute of Computational Aerodynamics, China Aerodynamics Research and Development Center, Mianyang 621000, Sichuan, China

Published online: 04 September 2025

computational parameter domain to the physical domain; however, it faces challenges in accommodating complex topologies. The region decomposition method, also known as multi-block structured mesh generation [5, 6], partitions intricate geometries into topologically simpler subdomains for independent meshing, necessitating substantial expert intervention for optimal partitioning. PDE-based methods [7–9] generate high-quality, smooth, and orthogonal grids by solving governing partial differential equations that dictate grid line distribution, yet they incur significant computational overhead and encounter difficulties in automating boundary handling and internal feature resolution. Overall, these conventional techniques often struggle to simultaneously optimize automation efficiency and mesh quality in complex geometric models prevalent in aerospace engineering and advanced manufacturing sectors. Consequently, substantial manual input remains essential for tasks such as geometric cleanup, domain partitioning, singularity point placement, mesh topology design, and density regulation.

In recent years, artificial intelligence technologies represented by deep learning have made breakthrough progress in various fields. Their powerful feature learning and nonlinear mapping capabilities have also brought new opportunities to solve the dilemmas of traditional grid generation methods. Lei et al. [10] conducted a systematic and comprehensive review of the emerging research field of intelligent grid generation, providing important references and guidance for research in this area. Neural network-based mesh generation methods, such as Convolutional Neural Networks (CNNs) [11], Graph Neural Networks (GNNs) [12], Physics-Informed Neural Networks (PINNs) [13, 14], and Generative Adversarial Networks (GANs) [15], learn the mapping relationship between geometric features and mesh topology in a data-driven manner, achieving notable performance on specific geometric types. However, these methods often face significant challenges in terms of generalization. When the target geometric shape significantly differs from the training samples, the quality of the generated mesh often deteriorates sharply, and the mesh elements may experience severe deformation or topological errors. This limitation severely restricts the applicability of learning-based methods in practical engineering applications, especially when dealing with unknown or highly variable geometric shapes.

In this paper, we present an automatic block-structured mesh generation framework integrating template-guided deep reinforcement learning (DRL) and discrete Ricci flow techniques. The framework operates on a three-stage principle: First, leveraging the conformal invariance of discrete Ricci flow, complex three-dimensional surfaces are robustly parameterized onto a two-dimensional planar domain. Adaptive partitioning of the parametric domain is subsequently performed based on geometric distortion metrics derived from the parameterization process, decomposing intricate problems into computationally tractable subproblems. Second, within each parameterized subdomain, classical topological templates are introduced as structural priors, transforming the mesh generation task into a sequential decision-making process that couples template selection with parameter optimization. Finally, a deep reinforcement learning agent is trained to infer optimal decision policies, enabling automatic determination of critical template control parameters. This approach ensures topological correctness while facilitating the efficient generation of high-quality block-structured meshes. The main contributions of this paper include:

- An innovative hierarchical intelligent mesh generation framework (DRL-MeshGen) is proposed, which integrates the conformal properties of discrete Ricci flow and the decision-making capabilities of deep reinforcement learning to enhance automation and generalization for block-structured meshing of complex geometries.
- A novel method for automatic parametric domain partitioning is proposed, based on conformal parameterization via discrete Ricci flow and distortion field analysis. The method adaptively decomposes complex parametric domains into topologically simpler subregions according to surface intrinsic geometry, facilitating subsequent template-based mesh generation.
- A deep reinforcement learning decision model integrating topological template priors is developed. The model transforms block-structured mesh generation into sequential template parameter optimization, enabling agents to efficiently learn optimal meshing strategies under template constraints and addressing generalization challenges of traditional methods for unknown geometries.

The remainder of this paper is organized as follows. In Sect. 2, we briefly review the relevant work on mesh generation. In Sect. 3, we provide an overview of the overall workflow of the proposed DRL-MeshGen framework. In Sect. 4, we detail the conformal parameterization and partitioning method based on discrete Ricci flow. In Sect. 5, we elaborate on the template-prior-guided reinforcement learning framework. In Sect. 6, we present experimental results and analysis. Finally, we conclude the proposed method and discuss future research directions in Sect. 7.

2 Related work

Numerous studies have been conducted on structured mesh generation methods. However, a detailed survey of all existing works is beyond the scope of this paper, and we only review the most relevant approaches here.

Frame Field. The domain decomposition approach, as pioneered by Kowalski et al. [16], utilizes the principles of heat diffusion to compute the frame field within the computational domain. This facilitates segmentation through the extraction of singular points of the frame field. This methodology has been demonstrated to serve two primary functions. Firstly, it identifies singularities, and secondly, it ensures the smooth transfer of boundary frame data into the interior region. Fogg et al. [7] built upon this concept by propagating boundary frames into the interior according to varying tensor fields and by optimizing the energy functional during propagation to achieve an optimal frame configuration. Viertel et al. [17] proposed an integrated frame field design that incorporates Ginzburg-Landau theory. This approach employs the extended Merriman-Bence-Osher threshold dynamics method to minimize the Ginzburg-Landau energy functional. The result of this methodology is the production of a smooth, boundary-aligned frame field. In their seminal paper, Qi et al. [18] proposed a Riemannian frame field computation method on curved surfaces that utilizes a combined alignment strategy. This strategy has the advantageous effect of preserving boundary features and maintaining in-plane curvature. Lei and Zheng et al. [19, 20] applied Abel-Jacobi theory to automatically locate singular points, ensuring the Riemannian metric satisfies specific properties, thereby enabling geodesic tracking to automatically delineate sub-region boundaries. While the frame-based domain decomposition method yields effective discretization results in two-dimensional settings, its extension to three-dimensional domains introduces additional feature constraints and computational complexity. Furthermore, the incorporation of directional information during the solution process has been demonstrated to result in a reduction in both computational efficiency and accuracy.

Conformal Mapping. During the processes of mapping and inverse mapping, it is possible to guarantee the angles of the meshes can be guaranteed to the greatest extent. Consequently, the orthogonal meshes generated in the parametric domain can still maintain orthogonality after being inversely mapped back to the physical domain. Gu et al. [21] proposed the discrete holomorphic 1-form algorithm, and this parameterization is global in a certain sense, with the exception of a few points. Lévy, B., et al. [22] generated a quasiconformal mapping by solving the least squares approximation of the Cauchy-Riemann equations. Jin et al. [23] introduced a unified discrete Ricci flow algorithm framework, breaking through the limitations of traditional algorithms in dealing with topological structures and curvature designs. Mullen et al. [24] obtained the conformal parameterization under free boundaries based on the spectral method. Chern et al. [25] expressed it in the form based on quaternions and then used the quadratic convex Dirichlet energy to solve the quasiconformal transformation on the tetrahedron. Xu et al. [26] applied the Schwarz-Christoffel conformal mapping of multi-connected regions to the mesh construction of the global ocean model, improving the simulation accuracy of oceanic fluids. Vinhais, C., et al. [27] proposed a new method for generating polygonal meshes of bifurcated blood vessels based on the technique of conformal mapping. They improved the model by calculating local intensity features, enabling slight deformation of the blood vessel tree mesh for fine-tuning.

Deep Learning. Deep learning (DL) employs CNNs or GNNs [12] to automatically extract mesh features, evaluate the local mesh quality (e.g., smoothness, orthogonality, and uniformity of element distribution), and suggest targeted correction recommendations. As demonstrated in our previous research, significant progress has been made in the field of mesh quality discrimination [11, 28], as evidenced by the development of automatic generation methods [29, 30]. These findings have been extensively documented in the extant literature, with relevant citations provided for further reference. The fundamental concept underpinning this approach is the utilization of a data-driven methodology to facilitate the prediction, correction, and automated optimization of mesh quality. Zhang et al. [31] developed a surface structural mesh generation framework (MeshLink), which extended the support for mesh data and related algorithms through a mesh-based feature information framework (MFIF). MeshGPT [32] adopted a sequence-based method to generate novel and diverse mesh structures through autoregressive sampling. Expanding on their previous research [33], Yu et al. [34] introduced a novel algorithm that integrates deep learning with the generalized polycube method to generate high-quality hexahedral meshes. However, due to the inherent limitations of supervised learning in deep learning methods, these methods are unable to perform global exploration in the context of large-scale topological decision-making problems. Instead, they rely heavily on the expertise of specific users and a significant amount of manual interactions.

Reinforcement Learning. In contrast, the advantages of reinforcement learning (RL) in complex decision-making problems are becoming increasingly evident. The RL model optimizes the mesh generation strategy through continuous interaction with the environment, and is able to adaptively explore diverse combinations of mesh topology optimization and domain division. Lorsung et al. [35] developed a general DRL framework (MeshDQN) based on the deep Q-network of GNNs, which iteratively coarsens the mesh while retaining the calculation of target attributes. Pan et al. [36] formulated the mesh generation problem as a Markov decision process (MDP) problem and used a RL algorithm named “soft actor-critic" to automatically learn the action

strategy for generating meshes from experiments. Tong et al. [37] combined the advancing front method with a neural network and used a “policy network" to select reference vertices and update the front boundaries. Lim et al. [38] ensured, through the RL framework, that each point located on the surface of an object can propagate outward from the surface of the object along an optimal path and will never intersect with each other. Zhao et al. [39] proposed the DeepMesh method, which enhanced the ability to generate high-face artificial triangulated meshes by introducing an innovative autoregressive generation framework. The successful application of such methods in mesh generation has opened up new possibilities for the automatic generation of high-quality structural meshes.

3 Method overview

The DRL-MeshGen framework proposed in this paper aims to achieve automated and intelligent generation of block-structured meshes for complex three-dimensional surfaces. As illustrated in Fig. 1, its core workflow consists of three key stages: conformal parameterization and partitioning based on discrete Ricci flow, template-prior-guided deep reinforcement learning for planar mesh generation, and inverse mapping back to three-dimensional surfaces.

Stage 1: Conformal parameterization and Geometry-Aware partitioning. Given an input triangular mesh of a three-dimensional surface (see Fig. 1(a)), we first employ the discrete Ricci flow algorithm (in Sect. 4.1) to conformally parameterize it onto a two-dimensional planar domain (see Fig. 1(b)). During parameterization, mesh elements in different regions undergo varying degrees of stretching or compression. This geometric distortion (in Sect. 4.3.1) reflects the local curvature and feature variations of the original surface. Innovatively, we leverage this information by analyzing the vertex distortion field of parameterized mesh elements and extracting isocontours as partitioning boundaries (in Sect. 4.3.2 and Fig. 1(c)), thereby adaptively decomposing the complex parametric domain into multiple subregions with relatively simple topological structures and consistent geometric features.

Stage 2: Template-guided deep reinforcement learning for planar mesh generation. After obtaining the

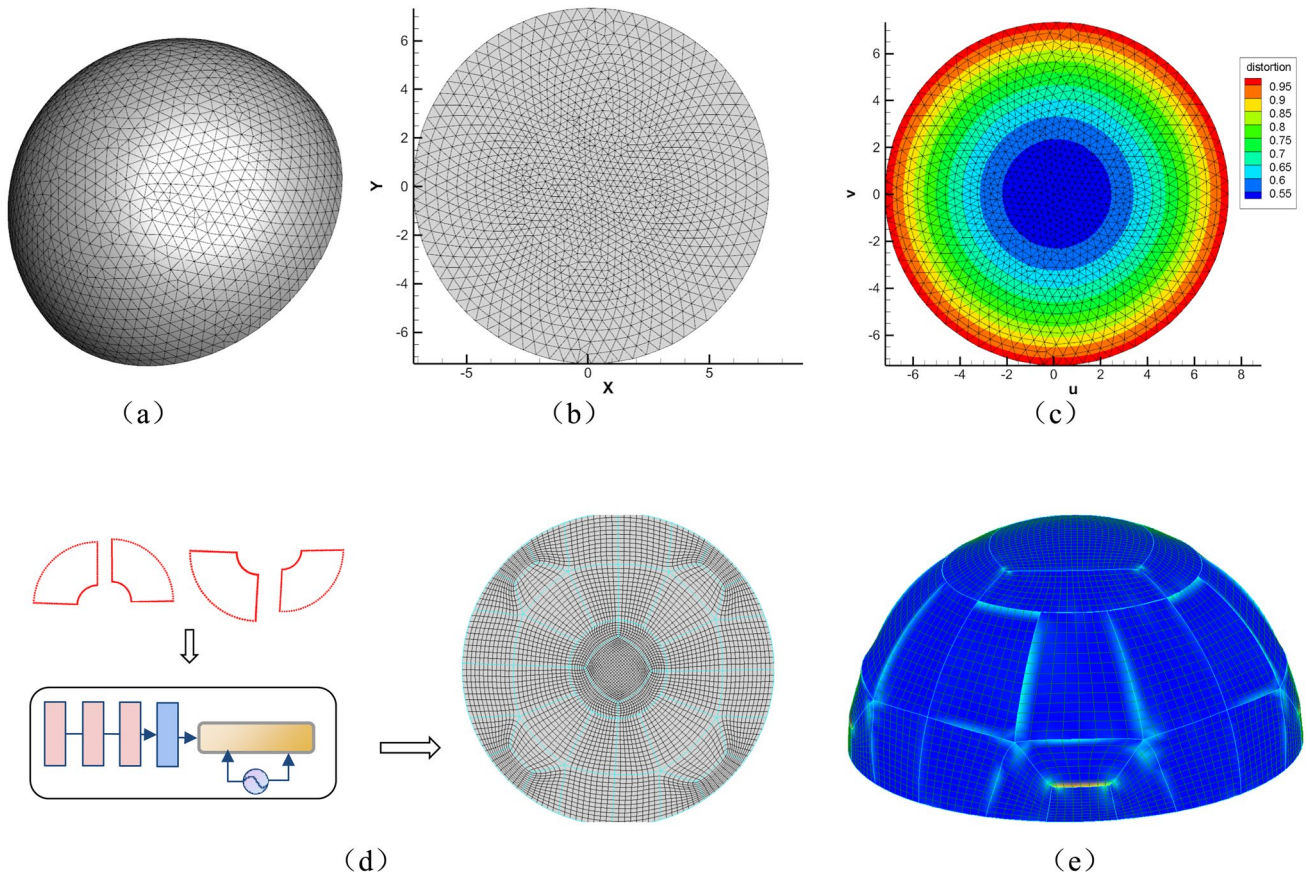

Fig. 1 The workflow of the proposed method. **a** The triangulated mesh of a given geometry, **b** conformal parameterization via discrete Ricci flow, **c** partitioning through contour extraction from parameterization distortion fields, **d** generation of 2D block-structured meshes in each partition using a template-prior-guided deep reinforcement learning framework, and **e** conformal inverse mapping to derive the final block-structured mesh on the surface

subregion partitioning of the parametric domain, an innovative approach integrating topological template priors with DRL is proposed (in Sect. 5). Classical block-structured mesh topological templates (see Fig. 7) are treated as powerful block-structured prior knowledge to constrain and guide the decision space of the DRL agent. Instead of freely connecting vertices and constructing elements from scratch, the task of reinforcement learning is transformed into a sequential decision-making problem for template parameter optimization (see Fig. 1(d)). This template-guided DRL approach significantly narrows the search space, improves learning efficiency, and ensures the topological correctness and structural regularity of generated meshes.

Stage 3: Mesh assembly and inverse mapping. Once the two-dimensional block-structured meshes for all parametric subregions are generated, they are assembled into a complete block-structured mesh of the parametric domain. Finally, leveraging the invertibility of the parameterization process, this planar block-structured mesh is transformed back to the original three-dimensional surface via conformal inverse mapping (see Fig. 1(e)), yielding the final three-dimensional surface block-structured mesh.

4 Conformal parametrization and partitioning

The great majority of curved surfaces are represented in discrete form in computer science and engineering. We can extend some key ideas from the continuous case, including metrics, curvatures, and conformal deformations, to the discrete case [40]. We use the rate of change in mesh density throughout the parameterization process to extract the partitioning boundaries and reduce the surface partitioning problem to a planar parametric domain based on the rigorous mathematical theory within the context of Euclidean geometry.

4.1 Discrete surface Ricci flow

Given that $\Omega(E, V, F)$ is a triangulated discrete surface, where V, E, and F represent the vertex, edge, and face sets, respectively. Each face f_{ijk} is a Euclidean triangle $[v_i, v_j, v_k]$, and its side lengths are $\{l_{jk}, l_{ki}, l_{ij}\}$ respectively. The interior angle of the triangle, denoted by θ_i^{jk}, and its side lengths satisfy the cosine law.

$$l_{jk}^2 = l_{ij}^2 + l_{ki}^2 - 2l_{ij}l_{ki}\cos\theta_i^{jk} \tag{1}$$

The discrete Gaussian curvature $K(v_i)$ of each vertex v_i can be expressed as:

$$K(v_i) = \begin{cases} 2\pi - \sum_{jk}\theta_i^{jk}, & v_i \notin \partial\Omega \\ \pi - \sum_{jk}\theta_i^{jk}, & v_i \in \partial\Omega \end{cases} \tag{2}$$

For each vertex v_i, its conformal factor u satisfies:

$$\frac{du(v_i)}{dt} = \overline{K}(v_i) - K(v_i, t) \tag{3}$$

among them, $\overline{K}(v_i)$ is the target Gaussian curvature at this point, t represents time, and the curvature $K(v_i, t)$ of each vertex changes with time. Solving the discrete Ricci energy is equivalent to solving a convex optimization problem, and the energy function is expressed as:

$$F(U) = -\int_0^u \sum_i (\overline{K}(v_i) - K(v_i))du_i \tag{4}$$

where $u = (u_1, u_2, \ldots, u_n)$.

The Hessian matrix of formula (4) is defined by the cotangent edge weights,

$$H = \frac{\partial^2 F(u^k)}{\partial u_i \partial u_j} = \frac{\partial K_i}{\partial u_j} = -\frac{1}{2}\omega_{ij} \tag{5}$$

The solution of the conformal factor can be reduced to solving a linear system,

$$H\delta u = \overline{K} - K \tag{6}$$

Newton's method can be employed to minimize the energy function $F(U)$ until the desired precision is achieved. This is indicated by the error tolerance, $max|K_i - K|$. The experimental results indicate that, under specific conditions, a high gain (defined as the ratio of precision to time) can be achieved. According to the experimental findings, this is achieved when the following condition is met: $\varepsilon = 1.0 \times 10^{-7}$.

For more details about the theory and algorithms of the discrete Ricci flow, please refer to [23, 41].

4.2 Conformal parametrization

Based on the content in Sect. 4.1, we propagate the target metric to the two-dimensional parametric domain. The side lengths of all triangular patches can be calculated from the target circle packing metric [42], providing important support for embedding the mesh elements into the parametric domain. For complex shapes, the side lengths of the mesh elements and the conformal factors at different vertices vary greatly. If any one element is selected as the initial embedding face, the embedding of the mesh elements will fail due to the existence of discrete errors (see Fig. 2).

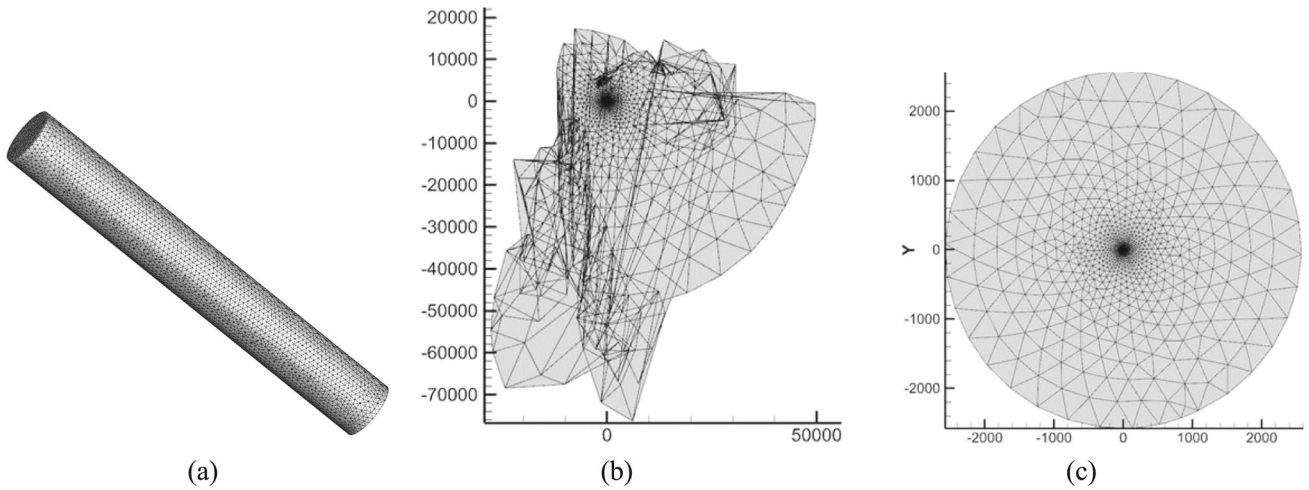

Fig. 2 Comparison of two different element embedding algorithms for the same shape. **a** The initially triangulated cylindrical model. **b** The tiling result of the arbitrary element embedding method. **c** The result of the method proposed in this paper

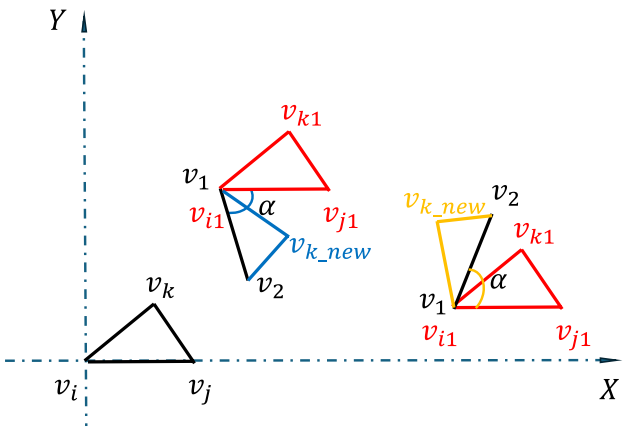

Fig. 3 Coordinate transformation for mesh element embedding in parametric domain. The embedding process involves two key geometric transformations. The target mesh element is first translated along vector Ov_1 and then rotated about vertex v_1 by angle α to preserve local connectivity with adjacent elements

To address these challenges, we propose a novel mesh embedding algorithm based on Riemannian manifold theory (see Algorithm 1). The computational workflow proceeds as follows:

- *Initial face selection.* The triangular face with minimal surface area is designated as the initial embedding face in the parametric domain, effectively mitigating the introduction of discretization errors.
- *Coordinate system initialization.* All mesh elements are positioned at the coordinate origin, with subsequent calculation of third-point coordinates to establish spatial relationships.
- *Iterative element processing.* Remaining elements are systematically processed via the breadth-first search (BFS) algorithm, with local geometric relationships preserved through rigid transformations (translation and rotation operations; see Fig. 3).
- *Topological update.* Vertex coordinate information undergoes dynamic updating throughout the embedding procedure to maintain topological consistency.

Input:
Initial face patch f_{012};
Vertex mapping from triangular faces to original surface $plane2Surface : \nu \rightarrow \mathbb{R}^3$.
Output: New mesh $newFaceMesh$ after parametric domain embedding.
Initialize marker $Mark = false$
Create face queue $tempFs$ and enqueue f_{012}
Establish mapping between the three vertices of the first embedded face patch and the parametric domain
Process remaining faces via breadth-first search (BFS)
while $tempFs$ is not empty **do**
  Dequeue the first element $tempF$
  if $tempF$ is marked **then**
    Continue and dequeue
  else
    Mark $tempF$
  end if
  for $halfedge$ in $tempF$ **do**
    if the dual half-edge is not a mesh boundary and the associated face f_{ijk} is unmarked **then**
      Extract three vertices and edge lengths of the adjacent face
      Construct rotation reference frame and compute basis vectors v_1v_2, v_1v_{j1}
      Calculate rotation angle α: $\cos\alpha = \frac{\vec{A}\cdot\vec{B}}{\|\vec{A}\|\|\vec{B}\|}$, $\sin\alpha = \frac{\|\vec{A}\times\vec{B}\|}{\|\vec{A}\|\|\vec{B}\|}$, and axis $s = v_1v_{j1} \times v_1v_2$
      if $s \geq 0$ **then**
        Point v_{k_new} lies to the right of v_1v_2, Counterclockwise rotation:
        $x_k = v_1[0] + (v_{k1}[0] - v_1[0]) \cdot \cos\alpha - (v_{k1}[1] - v_1[1]) \cdot \sin\alpha$
        $y_k = v_1[1] + (v_{k1}[0] - v_1[0]) \cdot \sin\alpha + (v_{k1}[1] - v_1[1]) \cdot \cos\alpha$
      else
        Point v_{k_new} lies to the left of v_1v_2, Clockwise rotation:
        $x_k = v_1[0] + (v_{k1}[0] - v_1[0]) \cdot \cos\alpha - (v_{k1}[1] - v_1[1]) \cdot (-\sin\alpha)$
        $y_k = v_1[1] + (v_{k1}[0] - v_1[0]) \cdot (-\sin\alpha) + (v_{k1}[1] - v_1[1]) \cdot \cos\alpha$
      end if
      Update coordinates of embedded face f_{ijk} to v_{k_new}
      Enqueue f_{ijk} into $tempFs$
    end if
  end for
end while

Algorithm 1 Riemannian manifold-based mesh embedding

4.3 Parametric domain partitioning

The present paper proposes a novel parameter domain partitioning method that utilizes the intrinsic geometric

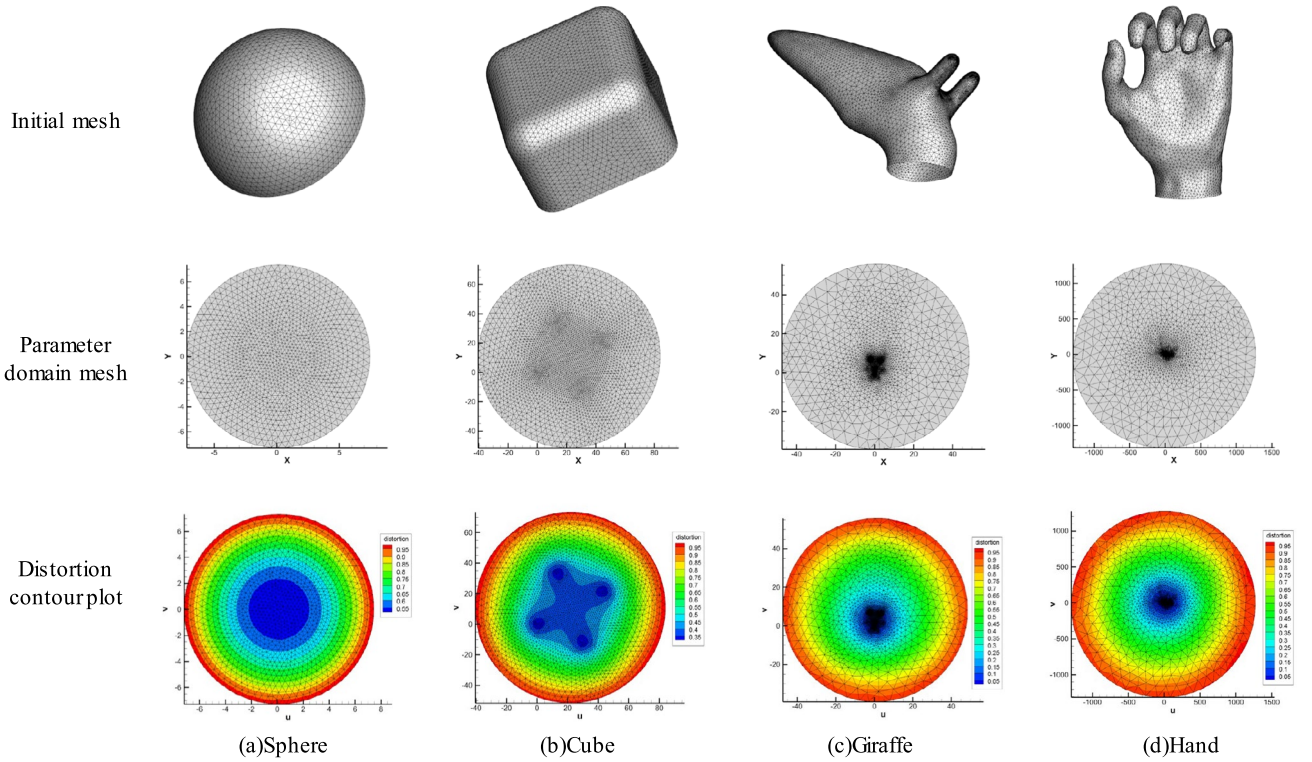

Fig. 4 Distribution of discrete vertex distortion across models after conformal parameterization. The first row shows the initial input meshes, the second row shows the meshes in the parametric domain, and the last row shows the contour cloud diagrams of the conformal distortion

properties of discrete Ricci flow. The method involves the use of isocontours within the parameter domain to serve as partition boundaries. The determination of these boundaries is based on the deformation degrees observed at mesh vertices before and after the application of conformal mapping, thereby facilitating the automated partitioning of the parameter domain mesh.

4.3.1 Distortion computation

The computation of conformal distortion generally relies on methodologies from differential geometry, complex analysis, and computer graphics, with specific approaches dictated by application contexts (e.g., continuous surfaces, discrete meshes, complex plane mappings) [43]. Since this study focuses on the parameterization of discrete surfaces, we utilize discrete vertex distortion to characterize conformal distortion.

Discrete vertex distortion is derived from geometric variations within vertex neighborhoods. We introduce the edge stretch ratio χ_e, defined as the ratio between the post-parameterization length L_{param} and original length L_{original} for each mesh edge e:

$$\chi_e = \frac{L_{\text{param}}(e)}{L_{\text{original}}(e)} \tag{7}$$

This metric quantitatively characterizes local geometric distortion during mesh parameterization. To ensure comparability across vertices with different degrees (e.g., $n_{v1} = 3$, $n_{v2} = 6$), we normalize and average the stretch ratios:

$$D(v) = \frac{1}{N(v)} \sum_{e \in N(v)} \frac{\chi_e}{\max(\chi_e)} \tag{8}$$

where $\max(\chi_e)$ represents the maximum edge stretch ratio, $N(v)$ denotes the vertex degree, and $D(v)$ constitutes the discrete vertex distortion. This approach prevents visualization artifacts from extreme values and eliminates degree-induced bias through normalization (see Fig. 4).

4.3.2 Domain partition

As established in Sect. 4.3.1, the distortion values at each vertex collectively form a density variation field across the parametric domain. The region-based decomposition approach extracts isocontours of cell density variation through local differential operations, which subsequently serve as partitioning boundaries. For practical applications,

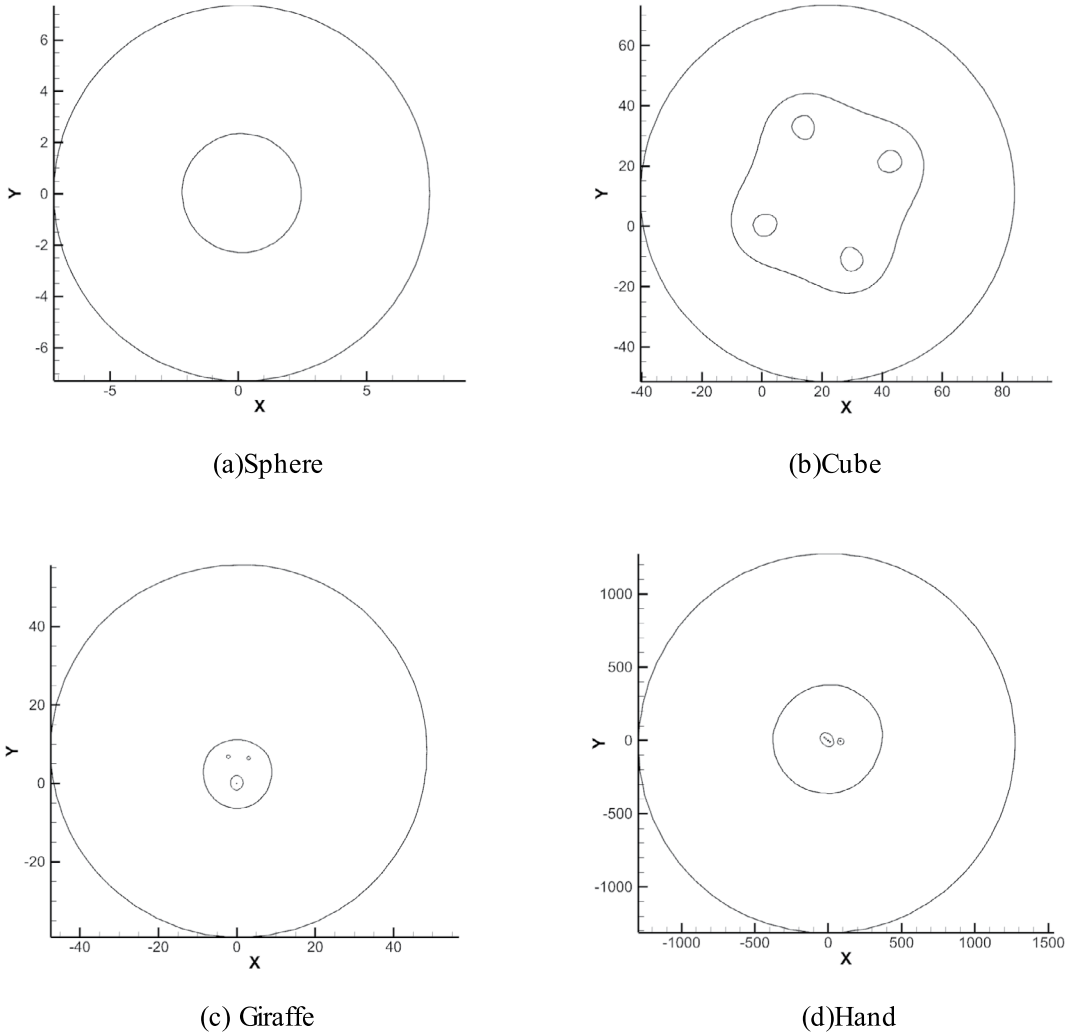

Fig. 5 Partitioning of the parametric domain. The partitioning in (**a**) consists of concentric rings, while the partitionings in (**b**), (**c**), and (**d**) are more complex, featuring multi-ring nesting

the proposed method demonstrates enhanced decomposition accuracy by accommodating significant spatial variations in mesh density across different domain regions (see Fig. 5).

Assume that an isometric parameter $f_{iso} \in (f_{min}, f_{max})$ is known. Extracting the set of contours $C = \{C_1, C_2, \ldots\}$ is the task we prioritize. As shown in Algorithm 2: Firstly, traverse the mesh elements in the parametric domain, find the edge intersection point P_{ij} that satisfies the isometric parameter f_{iso} through linear interpolation calculation, and classify the patterns of the edge intersection points.

$$P_{ij} = v_i + \frac{f_{iso} - f(v_i)}{f(v_j) - f(v_i)} \cdot (v_j - v_i) \tag{9}$$

where $f(v_i)$ and $f(v_j)$ are the distortion values corresponding to the vertices v_i and v_j. Secondly, connect the edge intersection points in each element to form contour segments. Thirdly, trace the path of the current contour to connect them into a closed curve in sequence. Finally, use Laplacian smoothing to smooth and optimize the generated curve, and perform enhancement processing on the mesh boundaries. Algorithm 2 illustrates the extraction process of all isocontours under a specific isovalue parameter.

For parameter domain decomposition of complex geometries, a set of isovalue parameters is typically required to partition the entire parameter domain into multiple subdomains through iterative application of Algorithm 2. Figure 6 demonstrates the decomposition results of the parameter

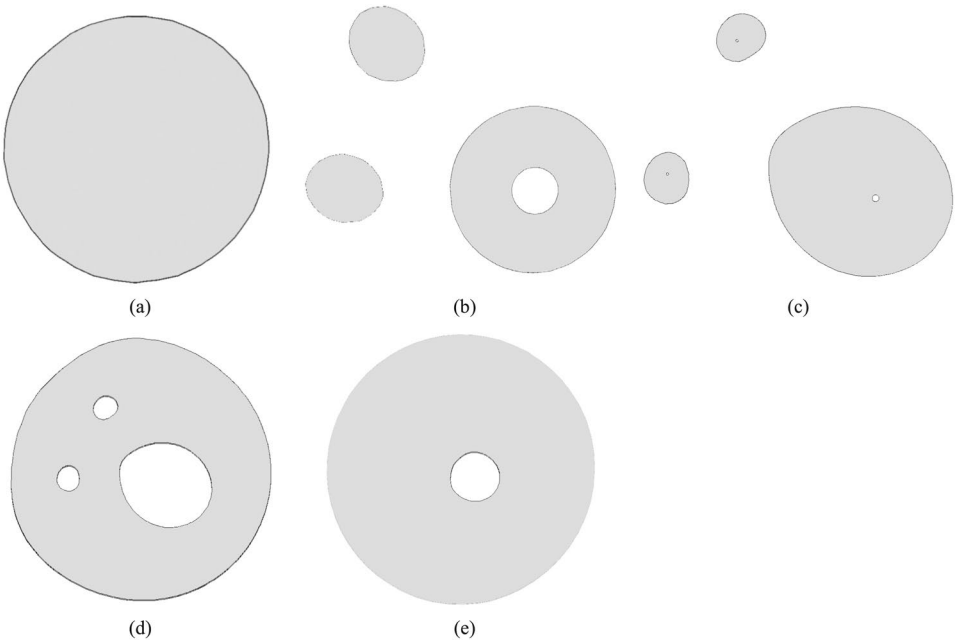

Fig. 6 Five partitions of the parameter domain for the giraffe model. **a** When the isovalue parameter $v_0 \in (v_{\min}, 0.001)$, the parameter domain is partitioned into a simply connected region. **b** When $v_0 \in (0.001, 0.003)$, the parameter domain is partitioned into 2 simply connected regions and 1 single-hole region. **c** When $v_0 \in (0.003, 0.05)$, the parameter domain is partitioned into 3 single-hole regions. **d** When $v_0 \in (0.05, 0.15)$, the parameter domain is partitioned into a multi-connected region with 3 independent holes. **e** When $v_0 \in (0.15, v_{\max})$, the parameter domain is partitioned into 1 single-hole region

domain for the giraffe model under different isovalue parameters v_0. To ensure continuity in the final global mesh generation, adjacent subdomains must maintain complete consistency in both the quantity and spatial distribution of discrete points along their shared boundaries, satisfying C^0 continuity. Independent boundary discretization for individual subdomains would violate this constraint, preventing successful merging into a valid global mesh. To address this challenge, our approach first performs unified boundary discretization for all subdomains (including both external and internal hole boundaries) based on the background mesh density in the parameter domain. This preprocessing step establishes consistent node distributions across all shared boundaries, effectively transforming the continuous geometric problem into a discrete topological problem.

Following regional decomposition and boundary discretization, the original complex geometric domain is converted into multiple subdomains with predetermined boundary node configurations, each suitable for independent processing. To robustly process these subdomains, we developed a reinforcement learning framework enhanced by template priors. This approach systematically manages different geometric configurations through a set of predefined topological templates.

Input:
Triangle list $T = \{t_1, t_2, ..., t_M\}$, where $t_j = (v_{a_j}, v_{b_j}, v_{c_j})$;
Vertex coordinate set $V = \{v_i = (x_i, y_i)\}_{i=1}^{N}$;
Distortion value set $f = \{f_i\}_{i=1}^{N}$;
Isovalue parameter f_{iso};
Triangle adjacency list A, where $A[j]$ contains indices of triangles adjacent to t_j.
Output: Contour set $C = \{C_1, C_2, ...\}$, where each C_k is a closed 2D coordinate sequence.

```
Initialize visited triangle set T(v) and empty contour list C
for Each triangle t ∈ T[j] do
    if t is visited then continue
    else
        Extract vertices (v0, v1, v2) and edge list e(t) = {[v0, v1], [v1, v2], [v2, v0]} of the triangle t
        for Edge e ∈ e(t) do
            Compute edge intersection P_e using Eq. 9
            if P_e ≠ 0 then push to temporary stack P(e)
            end if
        end for
        if P(e) contains 2 non-overlapping points then
            Initialize path C(e) = {P(e)[0], P(e)[1]} and add t_j to T(v)
            while true do
                for The adjacent triangle t_next ∈ A[j] of the triangle t do
                    if t_next ∉ T(v) then
                        Compute edge intersections for t_next
                        if The intersection point meets connection criteria then
                            Extend path C(e)
                        elseHandle branch
                        end if
                    end if
                end for
                Add t_next to T(v) and update current triangle t_i
                Repeat steps 13-23 until no new intersections found
            end while
            Convert C(e) to closed curve C_k and add to C
        end if
    end if
end for
```

Algorithm 2 Contour tracing

5 Template-prior guided reinforcement learning framework

5.1 Template priors for block-structured mesh

To construct high-quality block-structured meshes with regular topological connections in complex computational domains, researchers in related fields have developed a series of topological templates through long-term engineering practices and theoretical explorations, enabling efficient and robust topological partitioning for specific geometric configurations. Drawing on these practical experiences, we define topological template filling as a fundamental topological operation and integrate it into the reinforcement learning framework as prior knowledge. The geometric domain is decomposed into quadrilateral subregions through the process of parameterized instantiation. This process ensures that topological constraints are satisfied, while also ensuring consistent discrete point counts on opposite edges and shared boundaries between adjacent regions. Finally, algebraic methods are utilized to generate subregion meshes and assemble them into a global block-structured mesh.

5.1.1 Topological templates and parameterization

We define two fundamental topological operations: O-type and C-type variant topological template filling (see Fig. 7). The O-type topology template handles curved closed boundaries, while the C-type topology variant template addresses topological connections and point count transitions. These two topological templates form a minimal functionally complete set for constructing complex topologies in this paper.

The O-type topology template handles circular-like regions. For simply connected circular-like regions, the classical O-type topology template is typically employed in engineering applications for topology construction. Direct use of a single-block mapping method would cause boundary mesh lines to converge at corner points, resulting in severe mesh skewing and quality degradation. The O-type topology template decomposes the original region into five quadrilateral sub-blocks (one central block and four surrounding blocks) by introducing a quadrilateral core at the center of the region and radiating mesh lines outward, as illustrated in Fig. 7a. This significantly improves the mesh orthogonality at the boundaries, at the cost of introducing four trivalent singularities internally.

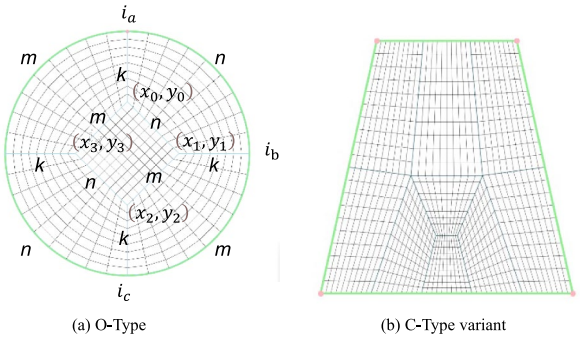

Fig. 7 Parameterized representations of the two topological templates. **a** O-type topology, introducing four singularities inside the computational domain to form a central core region. **b** C-type topology variant, introducing four singularities within the region, primarily used for transitioning points on long edges

The C-type topology variant template addresses point mismatch and transitions. After regional decomposition and boundary discretization, some subregions, although topologically quadrilateral, may exhibit a common geometric feature: a significant discrepancy in the number of mesh points between their two opposing arc edges (e.g., one derived from a high-curvature region of the outer boundary and the other from an internal isopleth or hole boundary). This point mismatch typically arises from substantial differences in geometric complexity and curvature variations across different regions of the computational domain. To tackle this issue, we introduce a variant of the C-type topology. Figure 7b presents the C-type topology variant adopted in this paper, which subdivides a quadrilateral region into 9 sub-blocks by introducing 4 singularities within the region. This allows the edge with more points to be decomposed into multiple segments: one segment directly corresponds to the edge with fewer points, while the remaining segments smoothly absorb the excess points through transition blocks. In addition to resolving point count transitions, the C-type topology variant can effectively control the direction and density of mesh lines in specific regions.

The topological templates are integrated into the reinforcement learning framework, and their structures are precisely described using a set of finite and quantifiable parameters. For instance, in the case of the O-type topology, given that the boundary of the computational domain has been discretized into L ordered and evenly spaced sampling points $P_0, P_1, \ldots, P_{L-1}$, the instantiation of an O-type template relies on the following key parameters:

1. **Boundary corner point selection:** Four representative index points (i_a, i_b, i_c, i_d) are selected from the L boundary discrete points as macroscopic boundary corner points. These corner points divide the original boundary into four segments with discrete point counts (edge lengths) of m_1, n_1, m_2, n_2, respectively, satisfying the constraint $m_1 + n_1 + m_2 + n_2 = L$. In a typical O-type symmetric structure, $m_1 = m_2 = m$ and $n_1 = n_2 = n$, simplifying the constraint to $2\,m + 2n = L$. The selection of corner points generally considers the geometric features of the computational

domain, such as curvature extreme points or geometric discontinuity points.

2. **Internal singular point localization and radial mesh layer determination:** The O-type topology introduces four trivalent singular points with coordinates (x_i, y_i) $(i = 0, 1, 2, 3)$ inside the computational domain. By connecting external boundary corner points to internal singular points and appropriately linking internal singular points, the original computational domain is accurately partitioned into five logical quadrilateral subregions. The precise positions of singular points and the radial mesh layer count k from boundaries to internal regions or between internal regions are key parameters to be optimized by the agent. The positions of singular points, as continuous variables, are not only interdependent but also constrained by previously selected boundary corner point positions. The value of radial mesh layer count k requires balancing the density of original boundary sampling points, singular point positions, and the desired mesh resolution.
3. **Adjustment of internal connection line types:** Given the potential complex curvature variations of the computational domain's physical boundaries, geometric adjustments to the lines connecting singular points and those linking singular points to boundary corner points (these lines form the subregion boundaries and are collectively referred to as internal edges hereafter) are necessary to ensure the quality of the final generated mesh. The rational selection of line types is crucial for guaranteeing that the subregion mesh points generated via algebraic methods exhibit good smoothness and orthogonality, while avoiding mesh distortion or inversion. Similar to the parameterized O-type topology template, the C-type topology variant template requires determining the positions of its internal singularities, the distribution of point counts on internal edges, and the line types of internal connecting lines.

5.1.2 Solution space scale

Following the definition of the template through the above quantifiable parameters, in theory, a brute-force exhaustive search of all parameter combinations could be used to find the optimal mesh partitioning scheme. However, the resulting solution space scale is extremely large, making exhaustive search computationally infeasible. We performed an order-of-magnitude estimation of the solution space scale (assuming the number of boundary points L reaches the hundreds order):

- *Boundary corner points and segmentation.* In the context of the constraint $2\,m + 2n = L$, the combinatorial order of magnitude for the values of m, n and the selection of corner point starting positions is approximately $O(L^2)$. When $L \approx 100$, the number of candidate combinations in this part is approximately 10^4.
- *Internal singular point localization.* The positions of singular points are continuous variables. When using discretization methods for optimization, assuming each singular point has N candidate positions in each coordinate dimension of the normalized space, the total number of candidate combinations for the four singular points (x_k, y_k) is $O\left(\left(N^2\right)^4\right) = O\left(N^8\right)$. As N increases, this term will dominate the complexity of the search space due to its exponential nature.
- *Radial mesh layer count.* The selection range of the radial mesh layer count k is relatively limited, and its contribution to the complexity of the search space can be approximated as a constant factor C_k.
- *Internal connection line types.* For the O-type topology, which has 8 critical internal edges (4 connecting singular points to form a central quadrilateral and 4 linking singular points to boundary corner points), assuming the line type of each internal edge can be selected from T predefined curve types, the total number of combinations for all internal edge line type selections is $O(T^8)$. When T ranges from 10 to 20, the order of magnitude of T^8 can reach 10^8 or higher.

Combining the above parts, the overall magnitude $N(T)$ of the parameter space can be estimated as:

$$N(T) \sim O(L^2) \times O(N^8) \times C_k \times O(T^8) \tag{10}$$

With $L = 100$, $N = 50$, $T = 10$, and C_k as a small constant, the total number of combinations can reach an astonishing order of magnitude of $10^4 \times (50^2)^4 \times 10^8 \approx 3.9 \times 10^{25}$, demonstrating the exponential explosion characteristic of this solution space.

Through in-depth analysis of the above solution space properties, we find it exhibits highly combinatorial, hierarchical structures, and mixed discrete-continuous characteristics. These features naturally make it an ideal application scenario for reinforcement learning (RL) methods. Therefore, we propose a novel approach for block-structured mesh generation in planar domains that deeply integrates template priors-guided reinforcement learning (TPG-RL). Compared with methods [36] that fully rely on RL to construct meshes point-by-point, TPG-RL significantly reduces the state-action space while leveraging the topological rationality constraints of templates to enhance learning efficiency and mesh quality.

5.2 Reinforcement learning framework

The TPG-RL method parameterizes templates to transform mesh generation into a parameter optimization problem. We formulate the template parameter optimization as a Markov Decision Processes (MDPs), where the RL agent aims to learn an optimal policy for sequential parameter determination. Instead of selecting all parameters simultaneously, the learning process unfolds through a series of ordered decisions. This section elaborates on the reward function design and DRL network architecture.

5.2.1 Reward function design

The goal of template-parameterized mesh generation is to identify template parameters yielding high-quality block-structured meshes. Accordingly, we design a terminal reward function R_{total}, which evaluates the agent upon completing the entire parameter selection sequence (i.e., determining all template parameters). The function performs integrated quality evaluation encompassing internal mesh properties (orthogonality and element shape quality) along with boundary consistency verification.

Mesh orthogonality. The ideal state of a block-structured mesh is that the mesh lines are as orthogonal as possible. Good orthogonality can reduce numerical discretization errors. In particular, benefiting from the conformal property of the initial Ricci flow mapping, an orthogonal mesh in the parametric domain can largely maintain its orthogonality when inversely mapped back to a three-dimensional surface, which is crucial for ensuring the quality of the final three-dimensional mesh. We evaluate the orthogonality $Q_{orth}(e)$ of each quadrilateral element e based on the deviation of its interior angles θ_i from $\frac{\pi}{2}$:

$$Q_{orth}(e) = 1 - \frac{2}{\pi} \min_{i=1}^{4} \left| \frac{\pi}{2} - \theta_i \right| \tag{11}$$

where θ_i are the four interior angles of the quadrilateral. For a perfectly orthogonal quadrilateral, $Q_{orth}(e) = 1$.

Element shape quality. Mesh elements should avoid excessive stretching, compression, or distortion to ensure the stability of numerical calculations. The metric based on the Jacobian determinant J is a commonly used method to measure element deformation. Calculate the Jacobian determinant J_i at the four nodes (or Gauss points) of element e:

$$Q_{jac}(e) = \frac{\min(J_1, J_2, J_3, J_4)}{\max(J_1, J_2, J_3, J_4)} \tag{12}$$

A value close to 1 indicates a regular element shape.

For the entire mesh M, the overall quality reward is defined as:

$$R_{quality}(M) = w_{orth} \frac{\sum_{e \in M} Q_{orth}(e)}{|M|} + w_{jac} \frac{\sum_{e \in M} Q_{jac}(e)}{|M|} \tag{13}$$

where $|M|$ is the number of mesh elements, w_{orth} and w_{jac} are weight coefficients that control the importance of orthogonality and Jacobian quality respectively.

Node matching accuracy. The nodes V_M^B on the mesh boundary should be as close as possible to the original boundary sampling points V_B. We use the negative mean squared error to measure the deviation:

$$R_{fidelity}(M) = -\frac{1}{|V_B|} \sum_{p_i \in V_B} \min_{p_j \in V_M^B} ||p_i - p_j||^2 \tag{14}$$

where V_B is the set of boundary sampling points, V_M^B is the set of mesh boundary nodes, and p_i and p_j are the coordinates of the corresponding points respectively. The value approaching zero indicates higher node matching accuracy.

The final terminal reward R_{total} is a weighted combination of the above rewards:

$$R_{total} = W_{quality} R_{quality}(M) + W_{fidelity} R_{fidelity}(M) \tag{15}$$

where $W_{quality}$ and $W_{fidelity}$ are top-level weights that control the overall trade-off between internal quality and boundary fidelity. All weights are hyperparameters adjusted and normalized through experiments to ensure the reward contributions from each component remain within reasonable ranges.

5.2.2 DRL network architecture

The value-based reinforcement learning algorithm (Double DQN) [44] is employed to implement and train the agent. Derived from the standard DQN, this algorithm effectively mitigates the problem of overestimating Q-values by decoupling action selection and action evaluation in target Q-value calculations, thereby enhancing the stability and performance of learning.

A deep neural network is employed to approximate the action-value function $Q(s, a; \varphi)$, where φ denotes the learnable parameters of the network. The Double Q-Network architecture is designed as shown in Fig. 8, and its main components include:

1. **Input layer:** The network receives the current state s_t belonging to the state space $\mathcal{S}$, which encapsulates all relevant information required for the agent to make decisions at decision time step t. This includes an

encoded representation of the computational domain's geometric features and the subset of template parameters determined up to the current time step. As input, it is formed by concatenating the graph neural network output h_G encoding the global boundary geometric features and the dynamic information encoding the current decision step and selected parameters.

2. **Main body of the network:** The input vector passes through a Multi-Layer Perceptron (MLP), which contains hidden layers as shown in the figure, and the ReLU non-linear activation function is used.
3. **Output head and action masking:** Since the action spaces vary at different decision-making steps (e.g., the size of the corner point selection space is $O(L^2)$ and the size of the singular point selection space is N^2), the output layer needs to adapt to this variation. We adopt the strategy of a shared output head with action masking:
   - The network outputs a fixed-size vector whose dimension equals the maximum number of possible discrete actions $|A|_{\max} = \max\left(O(L^2), N^2, k_{\max} - k_{\min} + 1, T_{\max}\right)$.
   - For the current decision-making step t, an action mask is generated according to its corresponding valid action subset $A(s_t)$. This mask sets the output Q-values corresponding to invalid actions to negative infinity (or a very small negative number).
   - When selecting an action, the agent chooses the action with the highest Q-value among valid actions only, i.e., $\arg\max_{a \in A(s_t)} Q_\phi(s_t, a)$.
4. **Double Q-Network mechanism:** To implement double DQN, we maintain two Q-networks with identical architectures but distinct parameters. The online network Q_ϕ is used to select actions according to the current policy at each time step, and its parameters ϕ are actively updated during training. The target network $\mathrm{Q}_{\phi'}$ computes the target Q-values in the Bellman equation, where its parameters ϕ' are not updated with every gradient descent step but instead copied from the online network's parameters ϕ after a certain number of training iterations. This delayed parameter update mechanism provides a more stable target for the learning process.

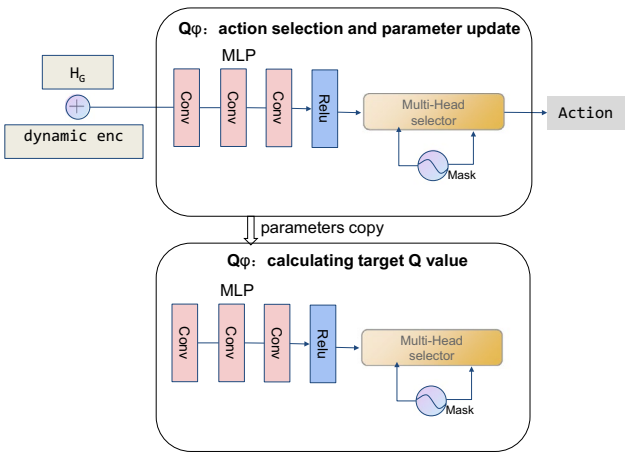

Fig. 8 Schematic diagram of the neural network-based Double Q-network architecture

Network parameters φ are updated by minimizing the mean squared Bellman error loss $L(\varphi)$. Specifically, for a transition (s, a, r, s', d) sampled from the experience replay buffer D, the loss function is defined as:

$$L(\varphi) = \mathbb{E}_{(s,a,r,s',d)\sim D}\left[\left(y - Q_\varphi(s,a)\right)^2\right] \tag{16}$$

Here, s is the current state, a is the executed action, r is the received reward, s' is the next state, and d is the episode termination flag. The target value y is computed as:

$$y = r + \gamma(1-d)Q_{\varphi'}(s', \mathrm{argmax}_{a'}\, Q_\varphi(s', a'))^\kappa \tag{17}$$

The action selection $(\mathrm{argmax}_{a'} Q_\varphi(s', a'))$ uses the online network Q_φ, while the value evaluation $(Q_{\varphi'}(\ldots))$ of the selected action uses the target network $Q_{\varphi'}$. This decoupling reduces the overestimation bias that may be caused by the same network selecting and evaluating the action.

The overall training process entails randomly sampling minibatches of experience data from the replay buffer $\mathcal{D}$, computing the loss $L(\phi)$ using this data, updating the online network parameters ϕ via a specific gradient descent algorithm, and periodically copying ϕ to the target network parameters ϕ'.

5.2.3 Mesh generation

Based on the DRL framework in the previous section, the mesh generation process for planar domains is primarily divided into two stages: agent training and inference.

The objective of the RL agent training stage is to train a DQN that can select optimal template parameters for a given geometric domain through decision-making. The training workflow is as follows: First, initialize the parameters of the online Q-network Q_ϕ and target Q-network $Q_{\phi'}$, as well as the experience replay buffer $\mathcal{D}$, select an initial template type (O-type or C-type) based on the methodology presented in [45]. Second, the entire training process involves multiple training episode iterations. For each episode starting from a computational domain instance, an initial state s_0 is constructed, and the agent selects actions a_t in the staged action space via an ε-greedy strategy to progressively determine

template parameters and transition to new states s_{t+1}. These transition experiences $(s_t, a_t, r_t, s_{t+1}, d_t)$ are stored in the buffer $\mathcal{D}$, where the immediate reward is 0 at intermediate steps, and only the terminal reward $R_{\text{total}}(M)$ is computed based on the final mesh quality at episode termination. Once all parameters are determined, the system instantiates the template, generates the complete mesh M using the TFI method, and evaluates its quality. Finally, during training, minibatches of experience data are periodically sampled from $\mathcal{D}$ to update the online network Q_ϕ via the double DQN algorithm and synchronize the parameters of the target network $Q_{\phi'}$ until the convergence criterion is met.

In the inference stage, the trained agent can automatically handle mesh generation tasks for new geometric domains. Given the boundary description of a target computational domain, the framework extracts geometric features h_G. Based on the trained Q-network Q_ϕ, the agent greedily selects the optimal action $a_t = \arg\max_{a'} Q_\phi(s_t, a')$ at each decision step to determine template parameters, generating a complete block-structured mesh that meets quality requirements through template instantiation and TFI algebraic filling. The entire inference process employs a deterministic policy, eliminating the need for exploration mechanisms and directly outputting mesh results with optimal parameter configurations.

6 Experimental results

6.1 Experimental setup

All experiments were conducted on a workstation equipped with an Intel Core i9-13900 CPU, an NVIDIA GeForce RTX 4090 GPU (24GB GDDR6X VRAM), and 64GB DDR5 RAM. The software environment is based on Python 3.9, with core dependency libraries including PyTorch 2.0 for neural network construction and training, OpenAI Gym 0.26 as the reinforcement learning environment interface, Stable-Baselines3 2.0 for implementing reinforcement learning algorithms, and our in-house developed mesh software NNW-GridStar [5] for data preprocessing and result visualization.

The TPG-RL agent has been trained using the Double DQN algorithm with the Adam optimizer (weight decay coefficient of 1×10^{-5}). The learning rate has been set to 1×10^{-4}, the discount factor $\gamma = 0.99$, and the target network is updated every 1000 steps. The experience replay buffer has a capacity of 50,000, with a batch size of 128. The exploration strategy employs the ε-greedy algorithm, where the initial value of ε is set to 1.0 and exponentially decays to a minimum of 0.05 at a decay rate of 0.999. The convergence criterion for training is defined as follows: training terminates when the growth rate of the average cumulative reward on the validation set is less than 1% over 50,000 consecutive training steps.

6.1.1 Dataset

To train and evaluate the TPG-RL agent, we constructed a corresponding dataset. The training and validation sets are derived from the boundaries of 2D subdomains obtained via conformal parameterization of approximately 100 engineering CAD models using Ricci flow. Data augmentation is performed through boundary point count perturbation (±20% of original points) and small-amplitude nonlinear geometric deformations (sine perturbation and local scaling), ultimately generating approximately 10,000 training samples and 1000 validation samples. These cover simply connected domains with diverse aspect ratios, boundary curvatures, and mild concavity/convexity features. The test sets consist of two parts: a standard test set (Test-Standard) containing 100 unseen 2D closed domains that share the same distribution as the training set, and a hard test set (Test-Hard) comprising 50 challenging simply connected subdomains derived from complex geometric configurations. The boundaries of these subdomains are generally more irregular or contain sharp internal angles. Examples of training/validation samples and hard test set samples are shown in Fig. 9. Additionally, for end-to-end evaluation, we conducted comprehensive testing on three representative 3D CAD models: a basic hemisphere, an intermediate-complexity giraffe model, and a highly detailed hand model. Each model was first parameterized into 2D computational domains. After generating block-structured meshes using our TPG-RL method, we reconstructed the original 3D surface meshes through inverse parameter mapping.

6.1.2 Evaluation metrics and methods

To objectively evaluate the model performance, this paper adopts the following quantitative evaluation metrics:

1. Average orthogonality: Measures the overall orthogonality quality of the mesh. We calculate the orthogonality quality $Q_{\text{orth}}(e)$ for each element and then take the average over all elements in the entire mesh. This metric ranges from [0, 1], with values closer to 1 indicating better mesh orthogonality.
2. Orthogonality compliance rate: Counts the proportion of elements in the mesh where the deviation of internal angles from 90° is less than 15°. This metric reflects the prevalence of high-quality elements.
3. Smoothness: Measured by calculating the average of the side length ratios of adjacent elements. Values closer

to 1 indicate smoother transitions in mesh size without abrupt changes.

4. Validity: The proportion of elements in the mesh with positive Jacobian determinants. A 100% validity rate is a prerequisite for generating usable meshes and serves as a baseline indicator of successful mesh generation.
5. Boundary fitting error: Uses the Mean Squared Error (MSE) to measure the deviation between the boundary nodes of the generated mesh and the sampled points on the original geometric boundary. It is calculated as the negative of the R_{fidelity} formula; values closer to 0 indicate a more accurate approximation of the original geometry by the mesh boundary.
6. Generation time: For our TPG-RL method, this refers to the time required for forward inference; for comparative methods, it denotes the total time needed to complete optimization and generation.

To comprehensively evaluate TPG-RL, we selected two representative categories of comparative methods: traditional parameter optimization methods based on gradient-based sequential quadratic programming (SQP) and heuristic genetic algorithm (GA). SQP is a classical gradient-based nonlinear optimization algorithm; in the experiments, we adopted a multi-start strategy to mitigate its sensitivity to initial values. GA is a heuristic global search algorithm, for which we configured a population size of 100 and evolutionary parameters with a maximum of 500 generations. The termination criteria for SQP and GA are set as 200 and 50,000 function evaluations, respectively, to guarantee algorithmic convergence. These traditional methods also directly optimize the template parameters defined in Sect. 5.1.1, using the total reward function R_{total} specified in Sect. 5.2.1 as the objective function. For our second comparative category, we selected Gmsh [46], a widely-used open-source mesh generation platform. To ensure a rigorous comparison, we implemented its standard 2D quasi-structured quadrilateral mesh generation algorithm [47] while leveraging advanced functionality (transfinite interpolation) for structured mesh generation.

6.2 Ablation experiments and training analysis

6.2.1 Ablation experiments on the reward function

As described in Sect. 5.2.1, the goal of structured mesh generation in the parameter domain is first to balance the two mutually restrictive objectives of internal mesh quality and boundary geometric fitting accuracy. Based on this understanding, in the design of the composite reward function, we first focused on two key components: the quality reward R_{quality} and the fidelity reward R_{fidelity}. Meanwhile, due to the inherent property of conformal mapping that preserves angles but cannot guarantee areas, the quality reward R_{quality} is further decomposed into two critical components: orthogonality (Q_{orth}) and Jacobian determinant (Q_{jac}). The former ensures the approximate perpendicularity of mesh lines, while the latter prevents element distortion and inversion. We set the top-level weights as $W_{\text{quality}} + W_{\text{fidelity}} = 1$ and the internal quality weights as $w_{\text{orth}} + w_{\text{jac}} = 1$. To verify the rationality of the selected weight coefficients and quantitatively evaluate the contribution of each component, we designed two sets of ablation experiments and finally selected the configuration: $W_{\text{quality}} = 0.8$, $W_{\text{fidelity}} = 0.2$, $w_{\text{orth}} = 0.6$, $w_{\text{jac}} = 0.4$.

1. Analysis of top-level reward components

To analyze the contribution of the top-level reward components R_{quality} (internal quality) and R_{fidelity} (boundary fitting), we trained two baseline agents with extreme weights and compared them with our full model:

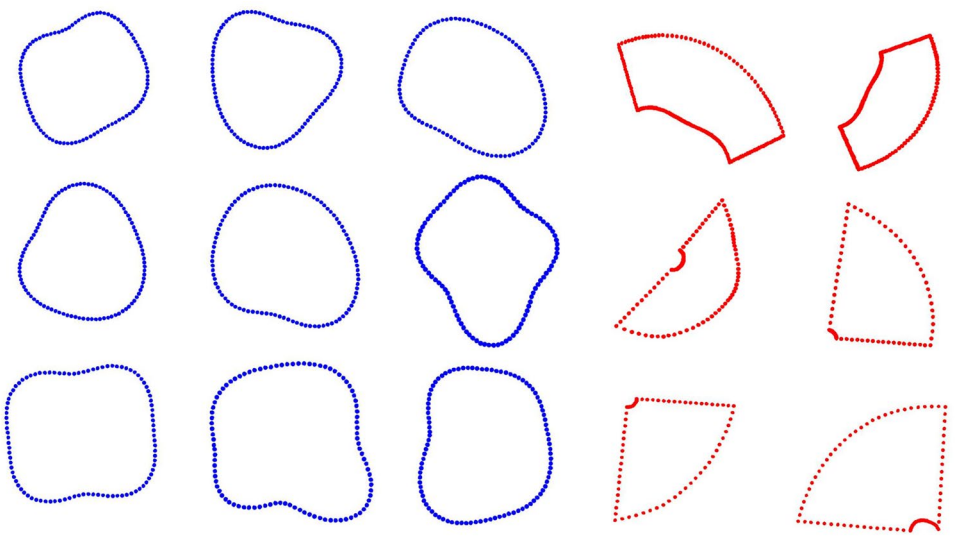

Fig. 9 Partial training and validation samples. The blue regions represent standard training and testing cases, while the red regions denote hard testing cases

- Quality-prioritized model ($W_{\text{quality}} = 1.0$, $W_{\text{fidelity}} = 0$). The agent focuses solely on maximizing the internal quality of the mesh.
- Fitting-prioritized model ($W_{\text{quality}} = 0, W_{\text{fidelity}} = 1.0$). The agent focuses solely on minimizing boundary geometric errors.
- Full model ($W_{\text{quality}} = 0.8$, $W_{\text{fidelity}} = 0.2$). The balanced configuration we ultimately adopted.

The experimental results presented in Table 1 demonstrate the necessity of balancing these two objectives. The quality-prioritized model, while generating meshes with regularly arranged internal elements, exhibits significant boundary fitting errors. In contrast, the fitting-prioritized model achieves high boundary conformity but at the expense of internal element quality, leading to the occurrence of invalid elements. Through appropriate weight balancing, the full model attains the optimal overall balance across all evaluation metrics.

2. Sensitivity analysis of internal quality weights

We now examine the two primary components comprising the internal quality reward R_{quality}: mesh orthogonality (Q_{orth}) and element shape quality (Q_{jac}). To analyze their trade-off relationship, we maintained constant top-level weights while systematically adjusting w_{orth} from 0.1 to 0.9, strictly enforcing the constraint $w_{\text{orth}} + w_{\text{jac}} = 1$. The impact of these weight variations was thoroughly evaluated using the Test-Standard benchmark dataset.

Figure 10 presents the Pareto frontier characterizing the relationship between average orthogonality quality and average element shape quality. The results demonstrate a fundamental trade-off between these competing metrics: any improvement in one metric inevitably compromises the other. Our chosen configuration ($w_{\text{orth}} = 0.6$, marked by a red star) corresponds to the Pareto frontier's inflection point where:

- Both quality metrics achieve their simultaneously highest attainable values.
- The marginal trade-off ratio becomes steepest (i.e., minor improvements in either metric require substantial sacrifices in the other)

This inflection point configuration represents the mathematically optimal balance between orthogonality preservation and element shape quality maintenance under the given constraints.

Table 1 Quantitative result comparison of ablation experiments on top-level reward weights

Configuration type	Boundary fitting error	Orthogonality	Validity(%)	Comprehensive score
Quality-prioritized model	0.89 ± 0.12	0.91 ± 0.02	100.0 ± 0.0	0.67
Fitting-prioritized model	0.15 ±0.03	0.45 ± 0.08	62.3 ± 8.7	0.41
Full model	0.28 ± 0.04	0.87 ± 0.03	100.0 ± 0.0	0.86

6.2.2 Analysis of the learning process

To illustrate the learning performance of the TPG-RL agent and the efficacy of the reward mechanism, we monitored mesh generation for a representative curved geometric domain in the training set (see Fig. 11(a)) across various training stages. Figure 11(b) shows the mesh generated by the agent for this domain in the early training stage (approximately 10% of total steps), where obvious defects in mesh quality are observed, accompanied by a low cumulative reward value of 0.003. In contrast, as depicted in Fig. 11(c), when the agent's policy approaches convergence, the mesh quality for the same domain is significantly improved: elements are more regularly arranged, orthogonality is enhanced, singular point positions are more reasonable, and the overall visual effect is superior, resulting in a significantly higher cumulative reward value of 0.66. It is noteworthy that the absence of flipped or overlapping elements is implicitly guaranteed by the Q_{jac} reward term. When the element Jacobian determinant is negative, Q_{jac} imposes a substantial penalty, effectively eliminating the generation of invalid meshes in the later stages of training. This renders post-processing corrections unnecessary for our method.

Table 2 provides further quantification of key mesh quality metrics for the samples depicted in Fig. 11 at the early training stage and post-convergence. The results show significant improvements across all metrics through reinforcement learning training, confirming that the TPG-RL framework optimizes parameter selection via reward function learning to progressively generate high-fidelity block-structured meshes.

6.3 Results analysis

6.3.1 Comparison with optimization methods

We compare TPG-RL with SQP and GA on both the Test-Standard and Test-Hard datasets. Table 3 summarizes the average quality metrics and generation times for each method.

Table 3 demonstrates that TPG-RL consistently matches or surpasses traditional optimization methods in all quality metrics, with superior robustness. Particularly on the more

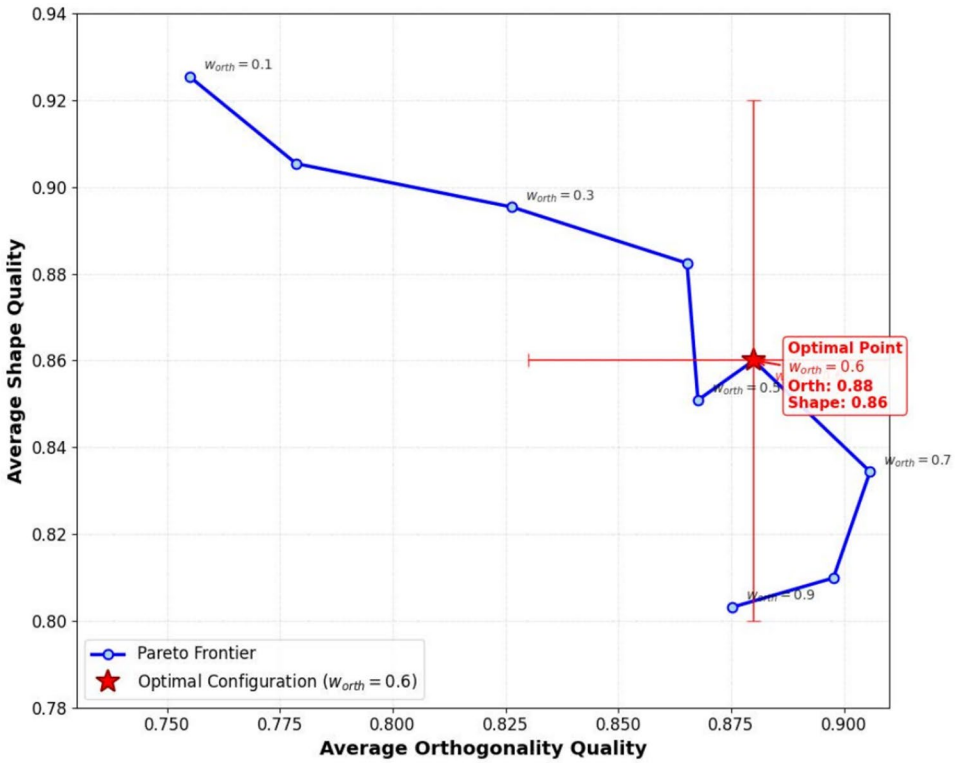

Fig. 10 Sensitivity analysis of weights for internal quality reward. The Pareto frontier varies with the average orthogonality and element shape quality as the weight w_{orth} changes. The red star indicates our finally selected configuration ($w_{\mathrm{orth}} = 0.6$)

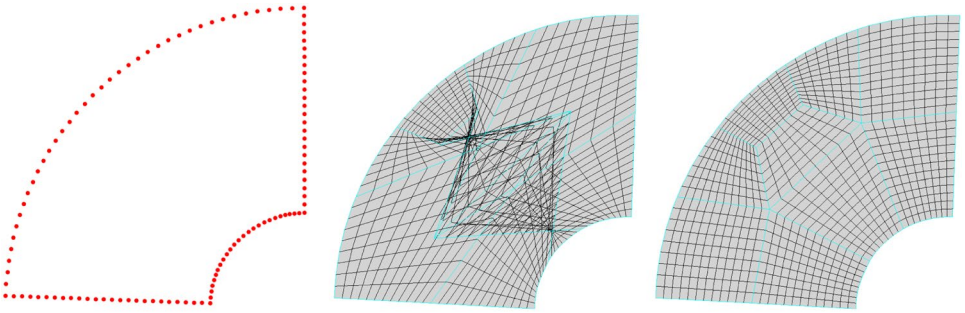

Fig. 11 Comparison of mesh optimization processes for a curved-feature geometric domain (C-type topology) during training. **a** Target geometric domain. **b** Mesh generated in the early training stage (approximately 10% of total steps). **c** Mesh generated after training convergence

challenging Test-Hard dataset, TPG-RL exhibits a smaller performance degradation, with a significantly lower standard deviation (± values) than SQP and GA, indicating stronger robustness and more stable results. Traditional methods are more prone to getting trapped in local optima on complex geometries, leading to larger performance fluctuations.

In terms of computational efficiency, the most notable advantage of TPG-RL lies in its online computational efficiency. After a one-time offline training, its average online inference time is only 25.7 ms. In contrast, SQP and GA require time-consuming iterative optimization for each new case, with average time costs being 1400 to 9500 times that of TPG-RL. Although TPG-RL incurs a higher one-time training cost, it exhibits an amortized efficiency advantage in scenarios involving large-scale geometry libraries or applications requiring near-real-time feedback.

Table 2 Comparison of key mesh quality metrics between the early and late training stages

Metric	Early Training (10% Steps)	Late Training (Convergence)
Orthogonality	0.62	0.87
Orthogonality threshold compliance rate (Deviation $< 15°$)	42%	89%
Smoothness (edge ratio)	1.48	1.13
Valid elements (%)	47%	100%
Cumulative reward (R_{total})	0.003	0.66

6.3.2 Comparison with Gmsh

In addition to comparing with traditional parameter optimization methods, we also conduct a comparison with the 2D quasi-structured mesh generation method of Gmsh, an

Table 3 Comparison of mesh quality and generation time between TPG-RL and traditional optimization methods (SQP, GA) on test sets

Test set	Method	Orthogonality	Smoothness	Shape quality	Boundary fit	Generation time
Test-Standard	TPG-RL	**0.88±0.05**	**0.93±0.04**	**0.86±0.06**	**0.97±0.02**	**25.7ms**
Test-Standard	SQP	0.87±0.08	0.91±0.06	0.84±0.07	0.95±0.04	35.6s
Test-Standard	GA	0.86±0.07	0.92±0.05	0.85±0.06	0.96±0.03	174.3s
Test-Hard	TPG-RL	**0.84±0.06**	**0.89±0.05**	**0.82±0.07**	**0.94±0.03**	**25.7ms**
Test-Hard	SQP	0.77±0.13	0.81±0.12	0.74±0.14	0.88±0.09	51.2s
Test-Hard	GA	0.80±0.11	0.85±0.08	0.78±0.10	0.90±0.07	243.8s

All quality metrics range from 0 to 1, with higher values indicating better quality. The computation time represents the average processing time per geometric domain. For TPG-RL, the reported time includes only inference time, excluding the one-off 48-hour training duration

Table 4 Quantitative performance comparison between TPG-RL and Gmsh on test sets

Test set	Method	Orthogonality	Smoothness	Shape quality	Validity(%)	Generation time(ms)
Test-Standard	TPG-RL	**0.88±0.05**	0.93±0.04	0.86±0.06	100	**25.7**
Test-Standard	Gmsh	0.85±0.07	**0.94±0.03**	**0.88±0.05**	100	45.3
Test-Hard	TPG-RL	**0.84±0.06**	**0.89±0.05**	0.82±0.07	100	**25.7**
Test-Hard	Gmsh	0.79±0.11	0.86±0.09	**0.83±0.08**	100	48.1

industrial-grade general-purpose mesh generation tool. The comparison primarily covers two dimensions: quantitative metrics and qualitative structures.

1. Quantitative analysis

We first conducted a quantitative evaluation of the mesh quality and efficiency generated by TPG-RL and Gmsh on both the Test-Standard and Test-Hard datasets. The evaluation results are summarized in Table 4.

Analysis of the data in Table 4 leads to the following conclusions:

- Orthogonality: TPG-RL significantly outperforms Gmsh in orthogonality metrics across both test sets. This advantage stems from the design of our reward function, which prioritizes orthogonality as a key optimization objective, enabling the reinforcement learning agent to successfully learn strategies that maximize this metric. This is critical for applications such as CFD, which demand high-quality boundary layer meshes.
- General quality and robustness: As a mature general-purpose tool, Gmsh exhibits excellent performance in smoothness and element shape quality due to its built-in smoothing and optimization algorithms, slightly outperforming TPG-RL on the standard test set. However, when handling the Test-Hard dataset–characterized by more irregular boundaries and sharp angles–TPG-RL demonstrates stronger robustness. Its quality metrics show a smaller degradation compared to Gmsh, with lower standard deviations, indicating superior adaptability and stability of TPG-RL for complex geometries.
- Generation efficiency: In terms of computational efficiency, after one-time training, TPG-RL achieves online inference speeds comparable to or even faster than the highly optimized Gmsh. Both methods operate at the millisecond level, far exceeding traditional iterative optimization approaches (e.g., SQP and GA). This confirms that our method is sufficiently efficient to rival industrial-grade software.

2. Topological structure and qualitative analysis

Beyond quantitative metrics, there is a fundamental difference in mesh topological structure between the two methods, which is crucial for specific applications. As shown in Fig. 12, by learning to select and optimize predefined topological templates (O-type/C-type), TPG-RL can generate strictly block-structured meshes with global consistency. In contrast, the meshes generated by Gmsh are quasi-structured. Although their local element quality is high, the mesh lines lack a global and controllable topological structure. Gmsh generates meshes through propagation and filling within the geometric domain, where the topological structure is dynamically determined by the algorithm based on local geometric features, making it difficult for users to achieve global control.

6.4 Surface mesh validation

To verify the applicability of TPG-RL in a complete 3D workflow, we applied it to the generation of structured surface meshes for complex 3D models. The workflow consists of the following steps. Firstly, parameterize the 3D surface into a 2D domain using Discrete Ricci Flow (see Sect. 4.2). Secondly, perform parameter domain partitioning and multi-connected region decomposition (see Sect. 4.3). For multi-connected subregions resulting from partitioning, we employed a rule-based topological decomposition strategy: this method decomposes multi-connected domains into simply connected quadrilateral subdomains by introducing virtual dividing lines. Specifically, dividing lines are extended orthogonally from each hole, then connected to

adjacent holes or extended to the outer boundary according to priority rules, and finally the generated polygonal subdomains undergo recursive quadrilateralization. Thirdly, apply TPG-RL to generate 2D meshes on each simply connected subdomain. Finally, map the 2D meshes back to the original 3D surface via inverse mapping.

Figure 13 presents the block-structured surface meshes generated by our method for a hemisphere model, a giraffe model, and a highly detailed hand model. For the hemisphere model, depicted as the first example in Fig. 13, which possesses a relatively simple geometry and topology, the integration of TPG-RL with the parameterization and decomposition strategies demonstrated commendable performance. It successfully produced high-quality surface meshes exhibiting the desired topological structure. The mesh lines not only maintain good orthogonality in 3D space but also achieve smooth transitions between different regions. This initially substantiates the feasibility and effectiveness of the proposed framework in handling 3D models characterized by significant curvature features.

Our method performs well even when tackling challenging cases such as the giraffe and hand models (the latter two examples in Fig. 13). As observed in Fig. 13c, the overall orthogonality of mesh elements and size uniformity (with the maximum ratio of side lengths of adjacent elements controlled below 4) basically meet the required standards.

The primary benefit of the method proposed in this paper lies in the integration of optimal conformal mapping and deep reinforcement learning. This approach is innovatively applied to the field of block-structured mesh generation, enabling automated mesh generation while introducing intelligent mesh generation techniques. We conducted a comparative analysis between this method and existing mature quasi-structured mesh generation technologies, with the results presented in Fig. 14. It is noteworthy that for the two given test models (giraffe and hand, as shown in Fig. 14a), our method can well meet the requirements of multi-block matching structured mesh generation, and the features at the fingertips are well preserved (see Fig. 14b). In contrast, Bommes et al. [48] solved the smooth frame field by means of mixed integers, generating quadrilateral meshes of high quality in regions with gentle curvature. However, when dealing with complex models with sharp curvature variations and multiple singularities, the local repositioning of singularities is time-consuming, and a global search strategy is lacking (see Fig. 14c). Gmsh [47] adopts the advancing method and template filling method for quasi-structured quadrilateral mesh generation, achieving good performance in terms of the minimum angle of generated mesh elements. Nevertheless, this method uses the boundary lines of geometric models as partition boundaries, leading to problems such as quadrilateral element degradation and feature loss in regions with sharp curvature variations (see Fig. 14d). As shown in Table 5, compared with other methods, the quality of meshes generated by our method (see Fig. 14) is significantly superior.

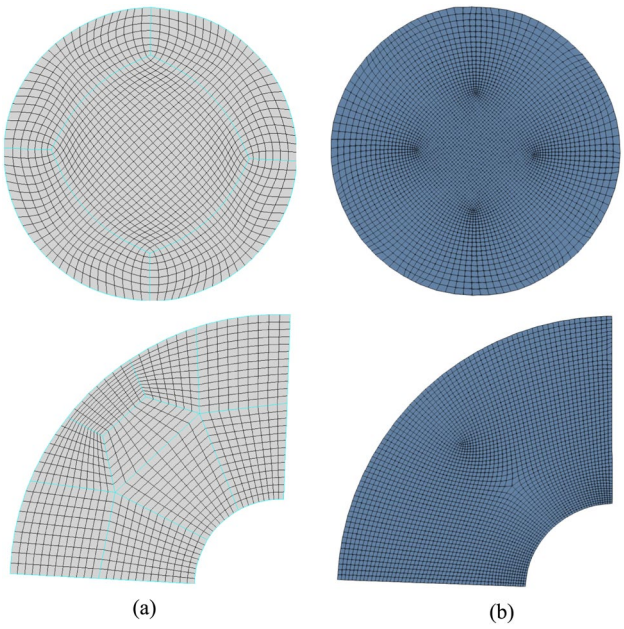

Fig. 12 Comparison of meshes for the same geometric domain. **a** O-type/C-type structured meshes generated by TPG-RL, featuring global topological consistency. **b** Quasi-structured quadrilateral meshes generated by Gmsh, with more isotropic element arrangement but lacking a global structure

We acknowledge that the current rule-based topological decomposition strategy is inherently a heuristic approach, which can only handle regular multi-connected cases. It lacks sufficient robustness when dealing with complex topologies (e.g., configurations with 6 or more irregularly distributed holes) and thus leaves room for improvement. Future research will focus on improving the decomposition and parameterization strategies for complex multi-connected regions and on augmenting the adaptability and robustness of the TPG-RL agent when subjected to more challenging boundary conditions and geometric constraints within these parameter domains. Such advancements would aim to achieve superior automated block-structured mesh generation across a broader spectrum of complex 3D applications.

6.5 Limitations and future work

Despite the demonstrated potential of DRL-MeshGen, there remain several limitations, which also point to directions for future research:

1. Topological limitations. The current framework primarily focuses on generating block-structured meshes for

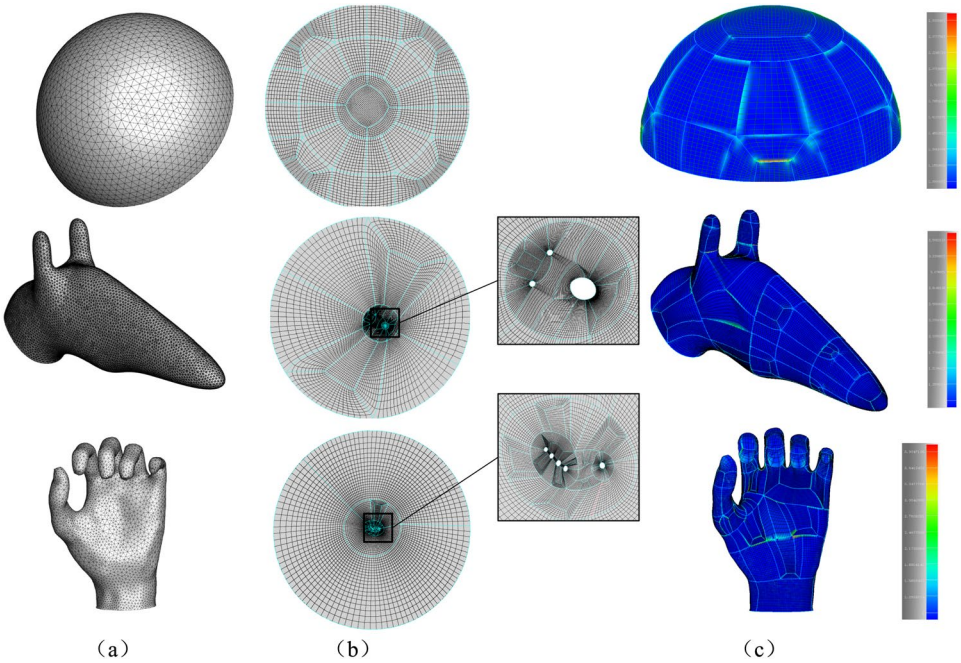

Fig. 13 Examples of TPG-RL applied to 3D block-structured surface mesh generation. **a** Input model; **b** Block structured mesh in the parameter domain generated by the TPG-RL agent, with details of internal multiple holes; **c** Length ratio of a certain dimension of the surface block-structured mesh generated after inverse mapping, where blue indicates ratios close to 1 and red indicates the maximum ratio

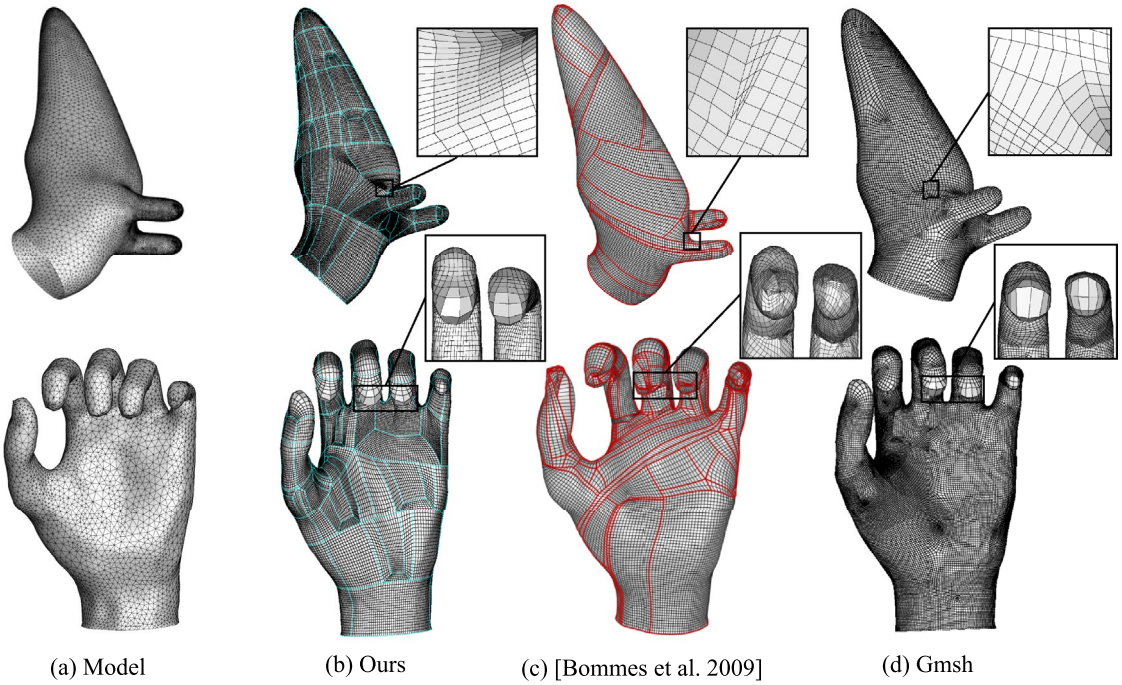

Fig. 14 Comparison of meshes generated by our method and existing mature quasi-structured methods. The first row shows the comparative results of the three methods on the giraffe model, and the second row presents those on the human hand model. Green lines represent the topological lines generated by our method, while red lines denote the streamlines generated by the method proposed by Bommes et al

non-closed surfaces. To handle closed surfaces (e.g., spheres with genus = 0), future work may introduce surface cutting algorithms (such as those based on fundamental group) prior to parameterization, converting them into topological structures compatible with the framework.

2. Singularity control. Under the current framework, the number and positions of singularities are autonomously determined by the agent based on the global reward function, lacking direct user control. Future research could explore incorporating penalty/guidance terms related to the number and positions of singularities into

Table 5 Comparison of quality statistics for quadrilateral meshes generated by our method and other quasi-structured methods in Fig. 14

Model	Method	Minimum angle	Maximum angle	Warp	Area ratio	Generation time(s)
Giraffe	Ours	14.58	**166.58**	**27.08**	**3.1**	**2.6**
Giraffe	Bommes	12.73	170.31	48.84	8.7	23.2
Giraffe	Gmsh	**31.4**	168.55	69.66	5.76	3.3
Hand	Ours	8.84	**173.11**	**51.6**	**6.05**	**5.9**
Hand	Bommes	5.1	176.26	174.54	41.62	49.4
Hand	Gmsh	**14.57**	176.38	161.69	8.86	53

the reward function, enabling explicit control over singularity layouts.

3. Local quality and post-processing. While global quality is excellent, some regions may still contain elements with suboptimal local quality. Future efforts may investigate integrating geometric smoothing techniques (e.g., geometry flow-based methods [49]) as a post-processing step or embedding them into the reward function to further enhance mesh quality.
4. Integration with engineering applications. To improve practicality in fields such as CFD, a key future research direction is to combine boundary layer mesh generation techniques [50, 51] with our framework, addressing the specialized mesh requirements of near-wall regions.
5. Exploration of reinforcement learning algorithms. We will evaluate more advanced RL algorithms (e.g., PPO, Rainbow DQN) to seek further improvements in training efficiency and final performance. Additionally, exploring more efficient geometric feature representations and refined reward mechanisms will remain ongoing optimization directions.

7 Conclusion

Compared with existing methods, the DRL-MeshGen framework exhibits significant advantages when dealing with complex geometric and topological features: Firstly, by transforming the problem of 3D surface mesh generation into the problem of block-structured mesh generation in a 2D parametric domain, it effectively reduces the complexity of the problem. Secondly, by abstracting the experience of classical topological templates into a discrete topological action space for reinforcement learning, the agent's learning objective shifts from directly generating meshes. Instead, upon selecting a topological action, it efficiently searches for the optimal instantiation configuration within the parameter space associated with that template–a space that would otherwise be prone to combinatorial explosion. Thirdly, through a meticulously designed multi-objective hierarchical reward mechanism, the system macroscopically guides the agent to balance the boundary geometric fitting degree and internal mesh quality, while microscopically further balancing the orthogonality and shape quality of elements.

Acknowledgements We extend our gratitude to the anonymous reviewers for their meticulous evaluation and for offering insightful and comprehensive feedback. We also gratefully acknowledge the Grid Generation Team at the Computational Aerodynamics Institute of CARDC for their essential support during manuscript preparation.

Author contributions L.Q. and Q.W. wrote the main manuscript text, coding, and prepared all the figures, R.M. developed and validated the code for conformal parameterization, J.Q., Y.L., J.X. and R.G. checked and modified the manuscript grammar, G.X. designed the manuscript framework and revised the overall content. Y.P. provided technical guidance for the manuscript and designed the experiments. All authors reviewed the manuscript.

Funding This work is partially supported by the National Key R & D Program of China under Grant No. 2023YFB3309100, "Pioneer" and "Leading Goose" R & D Program of Zhejiang Province (No. 2025C01086), the National Natural Science Foundation of China under Grant No. U22A2033.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Baker T (2005) Mesh generation: Art or science? Prog Aerosp Sci 41:29–63
2. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. Comput Methods Appl Mech Eng 194(39–41):4135–4195
3. Tong Y, Alliez P, Cohen-Steiner D, Desbrun M (2006) Designing quadrangulations with discrete harmonic forms. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006
4. Lee TY, Yen SW, Yeh IC (2008) Texture mapping with hard constraints using warping scheme. IEEE Trans Visual Comput Graphics 14(2):382–395
5. Lu F, Qi L, Jiang X, Liu G, Liu Y, Chen B, Pang Y, Hu X (2020) Nnw-gridstar: interactive structured mesh generation software for aircrafts. Adv Eng Softw 145:102803
6. Chuen A, Hosseini SS, Jensen JC, Chan W (2023) Aerodynamic simulations for complex geometries using automatically generated structured overset meshes. AIAA AVIATION 2023 Forum

7. Fogg JH (2015) Automatic generation of multiblock decompositions for meshing. IEEE Trans Visual Comput Graphics 101(13):965–991
8. Georgiadis C, Reberol M, Remacle JF (2022) Indirect all-quadrilateral meshing based on bipartite topological labeling. Eng Comput 38(5):4731–4747
9. Zhang S, Xu G, Wu H, Gu R, Qi L, Pang Y (2024) Medial hex-meshing: high-quality all-hexahedral mesh generation based on medial mesh. Eng Comput 40(4):2537–2557
10. Lei N, Li Z, Xu Z, Li Y, Gu X (2023) What's the situation with intelligent mesh generation: A survey and perspectives. IEEE Trans Visual Comput Graphics 30(8):4997–5017
11. Chen X, Liu J, Pang Y, Chen J, Chi L, Gong C (2020) Developing a new mesh quality evaluation method based on convolutional neural network. Eng Appl Comput Fluid Mech 14(1):391–400
12. Wang X, Yue Q, Liu X (2024) Intelligent mesh generation for crack simulation using graph neural networks. Comput Struct 292:107234
13. Aygun A, Maulik R, Karakus A (2023) Physics-informed neural networks for mesh deformation with exact boundary enforcement. Eng Appl Artif Intell 125:106660
14. Chen X, Liu J, Yan J, Wang Z, Gong C (2022) An improved structured mesh generation method based on physics-informed neural networks. arXiv preprint arXiv:2210.09546
15. Li T, Shi Y, Sun X, Wang J, Yin B (2021) Pcgan: Prediction-compensation generative adversarial network for meshes. IEEE Trans Circuits Syst Video Technol 32(7):4667–4679
16. Kowalski N, Ledoux F, Frey P (2015) Automatic domain partitioning for quadrilateral meshing with line constraints. Eng Comput 31:405–421
17. Viertel R, Osting B (2019) An approach to quad meshing based on harmonic cross-valued maps and the ginzburg-landau theory. SIAM J Sci Comput 41(1):452–479
18. Qi L, Qiu J, Xu G, Liu Y, Xu J, Gu R, Lu F, Pang Y (2024) Rff-meshing: a parallel and anisotropic quad-dominant mesh generation framework based on riemann frame field. Eng Comput 1–23
19. Lei N, Zheng X, Luo Z, Luo F, Gu X (2020) Quadrilateral mesh generation ii: Meromorphic quartic differentials and abel-jacobi condition. Comput Methods Appl Mech Eng 366:112980
20. Zheng X, Zhu Y, Chen W, Lei N, Luo Z, Gu X (2021) Quadrilateral mesh generation iii: Optimizing singularity configuration based on abel-jacobi theory. Comput Methods Appl Mech Eng 387:114146
21. Gu X, Yau S-T (2003) Global conformal surface parameterization. In: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 127–137
22. Lévy B, Petitjean S, Ray N, Maillot J (2023) Least squares conformal maps for automatic texture atlas generation. In: Seminal Graphics Papers: Pushing the Boundaries, Volume 2, pp. 193–202
23. Jin M, Kim J, Luo F, Gu X (2008) Discrete surface ricci flow. IEEE Trans Visual Comput Graph 14(5):1030–1043
24. Mullen P, Tong Y, Alliez P, Desbrun M (2008) Spectral conformal parameterization. In: Computer Graphics Forum, vol. 27, pp. 1487–1494. Wiley Online Library
25. Chern A, Pinkall U, Schröder P (2015) Close-to-conformal deformations of volumes. ACM Trans Graph (TOG) 34(4):1–13
26. Xu S, Wang B, Liu J (2015) On the use of schwarz-christoffel conformal mappings to the grid generation for global ocean models. Geosci Model Dev 8(10):3471–3485
27. Vinhais C, Kociński M, Materka A (2018) Centerline-radius polygonal-mesh modeling of bifurcated blood vessels in 3d images using conformal mapping. In: 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), pp. 180–185. IEEE
28. Chen X, Liu J, Gong C, Pang Y, Chen B (2020) An airfoil mesh quality criterion using deep neural networks. In: 2020 12th International Conference on Advanced Computational Intelligence (ICACI), pp. 536–541. IEEE
29. Chen X, Liu J, Zhang Q, Liu J, Wang Q, Deng L, Pang Y (2023) Developing a novel structured mesh generation method based on deep neural networks. Phys Fluids 35(9)
30. Chen X, Li T, Wan Q, He X, Gong C, Pang Y, Liu J (2022) Mgnet: a novel differential mesh generation method based on unsupervised neural networks. Eng Comput 38(5):4409–4421
31. Zhang H, Li H, Li N (2024) Meshlink: a surface structured mesh generation framework to facilitate automated data linkage. Adv Eng Softw 194:103661
32. Siddiqui Y, Alliegro A, Artemov A, Tommasi T, Sirigatti D, Rosov V, Dai A, Nießner M (2024) Meshgpt: Generating triangle meshes with decoder-only transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19615–19625
33. Yu Y, Wei X, Li A, Liu JG, He J, Zhang YJ (2022) Hexgen and hex2spline: polycube-based hexahedral mesh generation and spline modeling for isogeometric analysis applications in ls-dyna. Geometric challenges in isogeometric analysis. Springer, Cham, pp 333–363
34. Yu Y, Fang Y, Tong H, Zhang YJ (2025) Dl-polycube: deep learning enhanced generalized polycube method for high-quality hexahedral mesh generation and volumetric spline construction. Eng Comput 1–24
35. Lorsung C, Farimani AB (2022) Meshdqn: a deep reinforcement learning framework for improving meshes in computational fluid dynamics. arXiv preprint arXiv:2212.01428
36. Pan J, Huang J, Cheng G, Zeng Y (2023) Reinforcement learning for automatic quadrilateral mesh generation: A soft actor-critic approach. Neural Netw 157:288–304
37. Tong H, Qian K, Halilaj E, Zhang YJ (2023) Srl-assisted afm: generating planar unstructured quadrilateral meshes with supervised and reinforcement learning-assisted advancing front method. J Comput Sci 72:102109
38. Lim K, Lee S, Lee K, Yee K (2024) Mesh generation for flow analysis by using deep reinforcement learning. In: AIAA SCITECH 2024 Forum, p. 0382
39. Zhao R, Ye J, Wang Z, Liu G, Chen Y, Wang Y, Zhu J (2025) Deepmesh: Auto-regressive artist-mesh creation with reinforcement learning. arXiv preprint arXiv:2503.15265
40. Zhang M, Zeng W, Guo R, Luo F, Gu XD (2015) Survey on discrete surface ricci flow. J Comput Sci Technol 30:598–613
41. Yin X, Jin M, Luo F, Gu XD (2008) Discrete curvature flows for surfaces and 3-manifolds. LIX fall colloquium on emerging trends in visual computing. Springer, pp 38–74
42. Chow B, Luo F (2003) Combinatorial ricci flows on surfaces. J Differ Geom 63(1):97–129
43. Floater MS., Hormann K (2005) Surface parameterization: a tutorial and survey. Adv Multiresolution Geom Model 157–186
44. Hasselt HV, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: National Conference on Artificial Intelligence
45. Qiu B, Pang Y, Liu Y, Zhang X (2024) Reinforcement learning-driven adaptive region segmentation for structured mesh generation, 641–653
46. Geuzaine C, Remacle J-F (2009) Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer Meth Eng 79(11):1309–1331
47. Reberol M, Georgiadis C, Remacle J-F (2021) Quasi-structured quadrilateral meshing in gmsh – a robust pipeline for complex cad models. Computing Research Repository **abs/2103.04652**
48. Bommes D, Zimmer H, Kobbelt L (2009) Mixed-integer quadrangulation. ACM Trans Graph (TOG) 28(3):1–10

49. Zhang Y, Bajaj C, Xu G (2009) Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. Commun Numer Methods Eng 25(1):1–18
50. Yu Y, Zhang YJ, Takizawa K, Tezduyar TE, Sasaki T (2020) Anatomically realistic lumen motion representation in patient-specific space-time isogeometric flow analysis of coronary arteries with time-dependent medical-image data. Comput Mech 65:395–404
51. Lu F, Pang Y, Jiang X, Sun J, Huang Y, Wang Z, Ju J (2018) Automatic generation of structured multiblock boundary layer mesh for aircrafts. Adv Eng Softw 115:297–313