Computer-Aided Design 166 (2024) 103621

Contents lists available at ScienceDirect

Computer-Aided Design

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

Simultaneous Boundary and Interior Parameterization of Planar Domains Via Deep Learning

Zheng Zhan [a], Wenping Wang [b], Falai Chen [a,*]

[a] School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, PR China
[b] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843-3112, USA

ARTICLE INFO

Keywords:
Isogeometric analysis
Domain parameterization
Deep neural network

ABSTRACT

In isogeometric analysis (IGA), domain parameterization plays a crucial role as it significantly impacts the results of subsequent numerical analyses. Previous literature has often treated domain parameterization as two separate processes: boundary correspondence and interior parameterization. However, this division tends to decrease the quality of the final parameterization. To address this issue, this article proposes a learning-based method that simultaneously parameterizes the boundary and interior of a simply connected planar computational domain using a deep neural network. In our approach, we train a neural network to represent the inverse parameterization, which maps the computational domain to the parametric domain, based on the given boundary curve of the planar computational domain. The key component of our neural network is a Multilayer Perceptron (MLP), which takes the coordinates of a point in the computational domain as input and outputs the corresponding parametric values. The loss function of our method comprises three terms. First, it considers the area distortion and angular distortion of the parameterization, which are determined by the Jacobian matrix of the parameterization. Second, it incorporates the boundary difference, quantified by the Hausdorff distance between the boundary of the parametric domain and the image of the inverse mapping of the computational boundary. By optimizing this loss function, we achieve a robust parameterization. Once the network is trained, we employ a tensor-product B-spline function to fit the mapping from the parametric domain to the computational domain. To evaluate the effectiveness of our method, we conduct experiments on the MPEG-7 dataset. The experimental results demonstrate that our approach yields parameterization results with lower distortion and higher bijectivity ratio compared to state-of-the-art approaches.

1. Introduction

Isogeometric analysis (IGA) has gained significant popularity as an analysis tool that combines geometric modeling and engineering analysis [1]. An essential step in IGA is to construct a spline parameterization for the computational domain from its boundary. This process is known as domain parameterization (see Fig. 1). Over the past decade, extensive research has focused on domain parameterization. Most algorithms assume manual specification of boundary correspondence, thereby focusing solely on computing the interior parameterization. However, it has been observed that boundary correspondence strongly influences the quality of the interior parameterization. To address this, recent studies have introduced automatic algorithms based on optimal mass transport theory to compute boundary correspondences [2,3]. In [4], Liu et al. recognized that dividing domain parameterization into separate processes, namely boundary correspondence and interior parameterization, often leads to a decrease in the overall quality of the final parameterization. As a remedy, they proposed an approach that optimizes boundary correspondence and interior parameterization simultaneously. However, it should be noted that the parameterization results obtained using this method are not consistently satisfactory.

In this paper, we propose a deep learning based approach to optimize the boundary correspondence and the interior map simultaneously. In our algorithm, a neural network is used to represent the inverse parameterization that maps a computational domain C into a parametric domain $P = [0,1]^2$. The key component of our neural network is a Multilayer Perceptron (MLP), which takes the coordinates of a point in the computational domain as input and outputs the corresponding parametric values. To achieve this optimization, our approach incorporates a loss function comprising three essential components: the area distortion and the angle distortion of the interior map, and the difference between the image of the inverse map of the given boundary and the boundary of the parametric domain P. After the network is

* Corresponding author.
E-mail addresses: zz00822@mail.ustc.edu.cn (Z. Zhan), wenping@cs.hku.hk (W. Wang), chenfl@ustc.edu.cn (F. Chen).

https://doi.org/10.1016/j.cad.2023.103621
Received 4 June 2023; Received in revised form 17 September 2023; Accepted 23 September 2023
Available online 25 September 2023

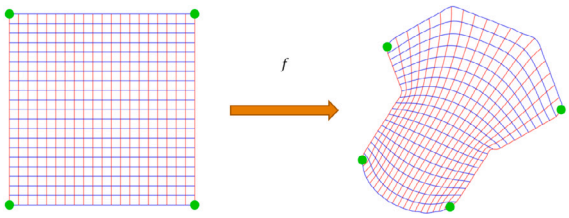


Fig. 1. A map from a parametric domain to a computational domain.

trained, we employ a tensor-product B-spline function to accurately fit the mapping from the parametric domain to the computational domain.

There are two challenges in realizing this approach. First, the computation of the difference between two boundaries is a time-consuming process. Second, the proper initialization of our network is crucial to maintain high parameterization quality; an improper choice of initialization can significantly diminish the effectiveness of the parameterization. To address these challenges, we propose two strategies. First, we observe that the one-sided Hausdorff distance from the inversely mapped boundary to the boundary of the parametric domain is easy to compute, and the Hausdorff distance is bounded by the one-sided Hausdorff distance with a factor of $\sqrt{2}$. This explains why we compute the inverse parameterization via the network. For the second challenge, we apply proper affine transformations on the computational domain and initialize the network with an identity map. By implementing these strategies, our method achieves high bijectivity ratio and low distortion for planar domain parameterization.

The remainder of the paper is organized as follows. Section 2 provides a review of related works on domain parameterization and geometric deep learning. In Section 3, we present a detailed description of our method, including the network architecture, theoretical analysis, the loss function, the pre-processing steps, the post-processing steps, etc. Section 4 presents a comparative analysis between our approach and the state-of-the-art parameterization methods [4–8] through experimental results. Finally, in Section 5, we conclude our work and discuss potential avenues for future research.

2. Related work

2.1. Domain parameterization

In the past decade, there have been a lot of research on domain parameterization. Early parameterization methods, such as Coons patches and the spring model [9], are linear methods and thus are simple and computationally efficient. However, these methods often result in non-bijective outcomes. Another popular category of methods are harmonic map-based methods, in which the parameterization problem is formulated as solving the Laplace equation with Dirichlet boundary conditions [10–13]. While harmonic methods are generally efficient, their parameterization quality may be inadequate for complex computational domains. To obtain high quality parameterization, various non-linear optimization-based methods [5,6,14–16] were introduced. These methods can produce parameterization with reasonable quality, yet they tend to be computationally expensive, especially in 3D case. To further enhance the parameterization quality, refinable splines were employed to represent the parametric equations of the computational domains [17–23]. When dealing with complex domains with non-zero genus, single-patch parameterization is not applicable. For these domains, multi-patch parameterization has become more common [19, 24–35]. However, it introduces two challenging issues: how to suitably partition the domain and how to ensure global smoothness of the parameterization [36–43]. In summary, computing high quality parameterization efficiently, especially for complex domains is still worthy of further investigation.

The aforementioned methods typically assume that the boundary map of the parameterization is provided, either manually or through automatic methods [2,3]. Consequently, domain parameterization is divided into two processes: boundary correspondence and interior domain parameterization that is tackled by the aforementioned methods. However, this division generally reduces the overall quality of the parameterization. Recently, Liu et al. [4] proposed an approach that simultaneously parameterizes boundary and interior domain. Up to now, this is the only method that optimizes the boundary correspondence and interior map simultaneously. Unfortunately, the parameterization quality achieved by this method is still unsatisfactory, as illustrated in Fig. 8 in Section 4, where some corner points are observed in flat or even concave areas. In this paper, we propose a deep neural network based method to simultaneously optimize the boundary and interior parameterization. Our method demonstrates superiority over the approach presented in [4], as well as the other state-of-the-art-methods [5–8] when incorporating boundary correspondence obtained through the automatic method in [2].

2.2. Deep neural network in geometry processing

In recent years, neural networks have gained widespread applications in various fields, including computer vision [44,45], natural language processing [46], numerical PDEs [47], medical imaging [48, 49] and many other fields. Deep learning has also made significant contributions to geometric processing, addressing a range of problems such as shape classification [50–52], shape correspondence [52–54], shape segmentation [55–57], shape completion [54,58], shape generation [51,59–62] and point data parameterization for polynomial curve approximation [63]. A survey on geometric deep learning can be found in [64–66].

Neural networks have also found applications in the problem of domain parameterization [8,67]. Similar to conventional methods, they only tackle the interior map with pre-given boundary correspondence. In [8], the parameterization of planar domains is formulated as the solution of a partial differential equation (PDE), then a physics informed neural network (PINN) is utilized to solve the PDE. Unfortunately, the authors claim that this method does not work for complex non-convex boundaries. In [67], a graph convolutional neural network is utilized for parameterizing surface patches represented by 3D point clouds. This method constructs a graph structure on the given point cloud, then utilizes a graph convolutional network to process the graph and predicts the barycentric coordinates on the parametric domain for each interior point. Combining the pre-given boundary correspondence and the network-predicted barycentric coordinates, the interior map can be computed.

In this work, we propose a new neural network based on MLP (Multilayer Perceptron) for planar domain parameterization. As a basic neural network structure, MLP is very suitable for fitting a general function or mapping, and it has been applied in various problems such as 3D point set classification and segmentation [68], 3D reconstruction [69], shape representation [70], etc. Thus it is quite natural to apply MLP in parameterizing computational domains.

3. Method

The objective of the current paper is to propose a deep learning-based method for simultaneous interior and boundary parameterization. Given the boundary of a computational domain, our method comprises the following steps: (1) training a neural network to obtain an inverse parameterization g from the computational domain C to the parametric domain $P = [0,1]^2$; (2) sampling a set of points Y in the computational domain C and computing the images $X = g(Y)$ in the parametric domain; (3) utilizing a tensor product B-spline to fit the inverse map of g based on the point sets X and Y, resulting in the final parameterization f from the parametric domain to the computational domain. In the following, we provide a detailed description of our method.

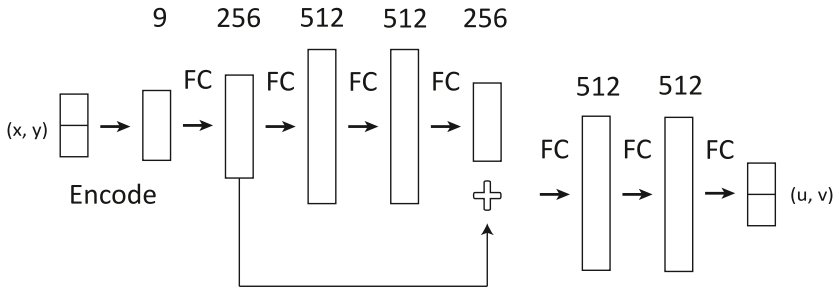


Fig. 2. Structure of the network $NetDP$. The input (x, y) is the coordinates of a point in the computational domain C, and the output (u, v) is the parametric values in the parametric domain P corresponding to the point. $Encode$ stands for the positional encoding operation of the input vector into a 9-dimensional vector $(x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2)$, FC refers to a fully connected layer, and the signature + stands for concatenation of two vectors.

3.1. Problem formulation

The problem of simultaneous optimization of the interior and boundary parameterization can be described as follows:

Formulation 1. *For a given computational domain C and the parametric domain $P = [0, 1]^2$, compute a parameterization f that maps P to C, which satisfies the following requirements:*

1. *The map f is bijective.*
2. *The map f has low area and angular distortion.*
3. *The boundary of P is mapped to the boundary of C, i.e., $f(\partial P) = \partial C$.*

For a given parameterization f, its bijectivity can be determined by examining the positivity of the determinant of the Jacobian matrix J_f. Furthermore, the area distortion and angular distortion of f can be quantified using J_f. In requirement 3, we aim to ensure that the Hausdorff distance $H(f(\partial P), \partial C)$ is smaller than a specified threshold. However, computing the Hausdorff distance $H(f(\partial P), \partial C)$ is a time-consuming task. Fortunately, as will be shown later, the Hausdorff distance $H(f^{-1}(\partial C), \partial P)$ can be controlled by the one-sided Hausdorff distance $h(f^{-1}(\partial C), \partial P)$ from $f^{-1}(\partial C)$ to ∂P, which is easy to compute. Thus, instead of computing the map f directly, we will calculate the inverse map of f. Hence, the new formulation of our problem becomes

Formulation 2. *For a given computational domain C and the parametric domain $P = [0, 1]^2$, compute an inverse parameterization g that maps C to P, which satisfies the following requirements:*

1. *The map g is bijective.*
2. *The map g has low area and angle distortion.*
4. *The one-sided Hausdorff distance $h(g(\partial C), \partial P)$ is smaller than a threshold.*

We translate the above formulation into an optimization problem.

Formulation 3. *For a given computational domain C and the parametric domain $P = [0, 1]^2$, compute an inverse parameterization g that maps C to P such that*

$$L := w_1 L_{area} + w_2 L_{angle} + w_3 L_{bound} \tag{1}$$

is minimized, where $w_1, w_2, w_3 \in \mathbf{R}$ are positive weights, $L_{area}, L_{angle}, L_{bound}$ are area distortion, angle distortion and boundary difference respectively. The details will be explained in Section 3.5.

In the above problem description, the objective function (1) consists of three terms, where $L_{angle} = 0$ results in a conformal map, while $L_{area} = 0$ leads to an area-preserving map. By adjusting the weights w_1 and w_2, we can control the optimization result to be more towards angle preserving or area preserving.

In this article, we will employ neural network training to learn the representation of the inverse parameterization g, using the loss function defined in (1). Once the neural network is trained, we obtain the inverse parameterization g. The final step involves fitting g^{-1} using a tensor product spline function to obtain the final parameterization. Further details will be discussed in Section 3.6.

3.2. Positional encoding

Positional encoding is a widely used technique in various architectures such as Transformer [71] and Neural Radiance Fields (NeRF) [69]. In Transformer architecture, positional encoding is utilized to provide some positional information of tokens in a sequence, enabling the network to capture the order of the input. In NeRF, positional encoding maps the inputs to a higher dimensional space using high-frequency functions, enhancing the model's ability to fit data with high-frequency variations. In this study, we employ positional encoding to map 2-D coordinates (x, y) into a 9-dimensional vector $(x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2)$ as the input, thereby improving the model's fitting capability.

3.3. Network structure

Our neural network used to approximate the inverse parameterization is a Multi-Layer Perceptron (MLP), whose structure is shown in detail in Fig. 2. The input consists of the coordinates (x, y) of a point within the computational domain C, which are encoded into a 9-dimensional vector $(x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2)$. The vector is then passed through 7 fully connected layers, and the output corresponds to the parametric values (u, v) within the parametric domain P, representing the map $g(x, y) = (u, v)$. Furthermore, a skip connection is incorporated within the network structure. This structure is implemented to shorten the propagation distance of gradients, enabling the network to converge more rapidly while maintaining a strong fitting capability. To ensure that the network represents a differentiable function over the computational domain, we define a differentiable function (2), referred to as $SmoothReLU$, as our activation function. And in our implementation, the constant β is set to be a relatively small value of 0.1. The training process is unsupervised and utilizes the loss function defined in (1).

$$SmoothReLU(x) = \begin{cases} 0, & x \le 0 \\ x^2/(2\beta), & 0 < x \le \beta \\ x - \beta/2, & x > \beta \end{cases} \tag{2}$$

After performing some preprocessing on the computational domain (refer to Section 3.6 for specific details), we initialize the network with an affine map and randomly select a set of points X ($\#X = 1000$ in our implementation) within C as the inputs in each iteration. The point set X is changed in each iteration, which helps to get a smooth result and avoid network overfitting. The network is trained until it converges or

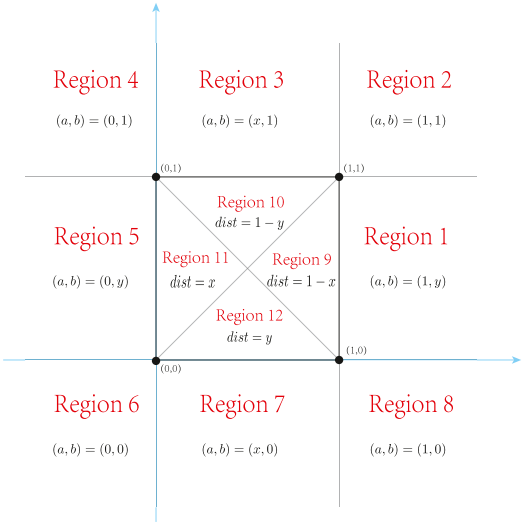


Fig. 3. Illustration of Lemma 1. The space $\mathbb{R}^2$ is divided into 12 regions. If the point (x, y) lies in the 4 regions inside P, its distance to the boundary ∂P is shown in the figure. If the point (x, y) lies outside P, its nearest point (a, b) on the boundary ∂P is given.

reaches the maximum number of iterations of 5000. For convenience, we use the notation $NetDP$ to refer to the network in the following discussions.

3.4. Hausdorff distance

To measure the difference between two boundaries, we employ Hausdorff distance.

Definition 1. Given two point sets A and B, the one-sided Hausdorff distance from A to B is defined as

$$h(A, B) = \max_{\mathbf{x}\in A}(dist(\mathbf{x}, B)) = \max_{\mathbf{x}\in A}(\min_{\mathbf{y}\in B}(dist(\mathbf{x}, \mathbf{y}))), \tag{3}$$

and the Hausdorff distance between A and B is defined by

$$H(A, B) = \max(h(A, B), h(B, A)), \tag{4}$$

where $dist(\cdot, \cdot)$ refers to the distance between two points or the distance from a point to a point set.

Let g be the map from the computational domain C to the parametric domain P. To measure the difference of $g(\partial C)$ and ∂P using Hausdorff distance, we introduce two lemmas at first.

Lemma 1. *For any point $(x, y) \in \mathbb{R}^2$ and the parametric domain $P = [0, 1]^2$, we have*

$$dist((x, y), \partial P) = \begin{cases} 1 - x, & (x, y) \in [0, 1]^2,\ y < x,\ y \geq 1 - x \\ 1 - y, & (x, y) \in [0, 1]^2,\ y \geq x,\ y \geq 1 - x \\ x, & (x, y) \in [0, 1]^2,\ y \geq x,\ y < 1 - x \\ y, & (x, y) \in [0, 1]^2,\ y < x,\ y < 1 - x \\ \sqrt{(x - a)^2 + (y - b)^2}, & (x, y) \notin [0, 1]^2 \end{cases} \tag{5}$$

Here $a = \max(\min(x, 1), 0)$, $b = \max(\min(y, 1), 0)$ and (a, b) is the nearest point of (x, y) on ∂P.

Proof. straightforward calculation. The details are illustrated in Fig. 3. □

Lemma 2. *The inequality $H(B, \partial P) \leq \sqrt{2}h(B, \partial P)$ holds for any boundary B of a simply connected domain that contains $(0.5, 0.5)$ as its inner point.*

Proof. By Definition 1, $H(B, \partial P) = \max(h(B, \partial P), h(\partial P, B))$. Thus, we only need to prove

$$h(\partial P, B) = \max_{\mathbf{x}\in\partial P} dist(\mathbf{x}, B) \leq \sqrt{2}h(B, \partial P)$$

or equivalently

$$dist(\mathbf{x}, B) \leq \sqrt{2}h(B, \partial P), \qquad \forall \mathbf{x} \in \partial P$$

For the cases illustrated in Figs. 4(a) and 4(c), for any point $A \in \partial P$, let O be the center point of P, and let the point A_1 be the intersection of the half line OA and the boundary B, and A_2 be the nearest point of A_1 on ∂P. Then we have

$$dist(A, B) \leq dist(A, A_1) \leq \sqrt{2}dist(A_1, A_2) \leq \sqrt{2}h(B, \partial P).$$

Similarly, for the cases illustrated in Fig. 4(b), we still have

$$dist(A, A_1) \leq \sqrt{2}dist(A_1, A_2) \leq \sqrt{2}dist(A_1, A_3) \leq \sqrt{2}h(B, \partial P). \quad \square$$

3.5. Loss function

The loss function of our network comprises three terms—area distortion, angular distortion, and boundary difference, as defined in (1) in Section 3.1. In the following, we will explain how these three terms are computed.

Let g be a differentiable map from $\mathbb{R}^2$ to $\mathbb{R}^2$ representing the inverse parameterization. We use J to denote the Jacobian matrix of g and $|J|$ to denote its determinant. Similarly, $J(\mathbf{p})$ and $|J(\mathbf{p})|$ represent the values of J and $|J|$ at a specific point $\mathbf{p}$. The area distortion and angular distortion can be characterized by the Jacobian of the map g.

Area distortion. Since the area of the computational domain is scaled to 1 in the pre-processing step according to Section 3.6, we define the area distortion of the map g at point $\mathbf{p}$ as

$$L_{area}(\mathbf{p}) = \left(|J(\mathbf{p})| + \frac{1}{|J(\mathbf{p})|} - 2\right)^2. \tag{6}$$

Notice that $L_{area}(\mathbf{p})$ attains a minimum value of 0 if and only if $|J(\mathbf{p})| = 1$. When the absolute value of $|J(\mathbf{p})|$ is small or large, $L_{area}(\mathbf{p})$ increases. Since our network is initialized with a linear map, the initial $|J(\mathbf{p})|$ is a positive value close to 1. During the optimization process, $|J(\mathbf{p})|$ will not become negative, because the loss $L_{area}(\mathbf{p})$ becomes very large when $|J(\mathbf{p})|$ approaches 0.

The area distortion over a domain C is defined by

$$L_{area}(C) = \int_C L_{area}(\mathbf{p})\, d\mathbf{p}. \tag{7}$$

In discrete form

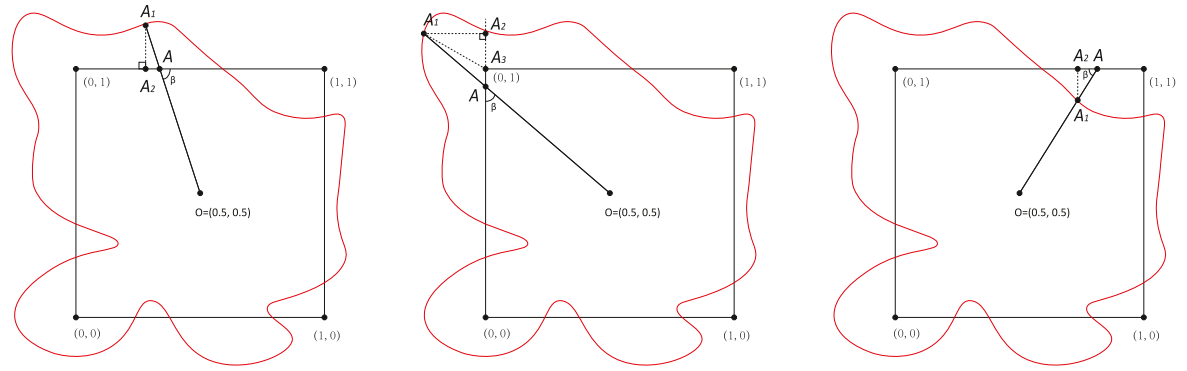
$$L_{area} = \frac{1}{m}\sum_{i=1}^{m} L_{area}(\mathbf{p}_i), \tag{8}$$

where $\{\mathbf{p}_i\}_{i=1}^m$ are a set of points uniformly sampled in C.

Angular distortion. The map g can be regarded as a complex function $g(z)$ where $z = x + y\sqrt{-1}$. The angular distortion of g can be measured by the Beltrami coefficient of the complex function $g(z)$, defined as $\mu(z) = g_{\bar{z}}/g_z$ [6]. In this expression, $\bar{z} = x - y\sqrt{-1}$ is the conjugate of z, $g_z = \frac{1}{2}(g_x - \sqrt{-1}g_y)$, $g_{\bar{z}} = \frac{1}{2}(g_x + \sqrt{-1}g_y)$ are complex derivates. It is straightforward to calculate that

$$|\mu(z)|^2 = \frac{\|J(\mathbf{p})\|_F^2 - 2|J(\mathbf{p})|}{\|J(\mathbf{p})\|_F^2 + 2|J(\mathbf{p})|}, \tag{9}$$

where $\|\cdot\|_F$ stands for the Frobenius norm, and $\mathbf{p} = (x, y)$ corresponds to $z = x + y\sqrt{-1}$. Notice that $|\mu| < 1$ if and only if $|J| > 0$, indicating



(a) The cases that A_1 lies in the region 3 (the case that A_1 lies in the regions $1, 5, 7$ is similar).

(b) The case that A_1 lies in the region 4 (the case that A_1 lies in the regions $2, 6, 8$ is similar).

(c) The cases that A_1 lies in region 10 (the case that A_1 lies in the regions $9, 11, 12$ is similar).

Fig. 4. The illustration of Lemma 2 discussed in categories. The red curve is the boundary B of a simply connected domain that contains $(0.5, 0.5)$ as its inner point.

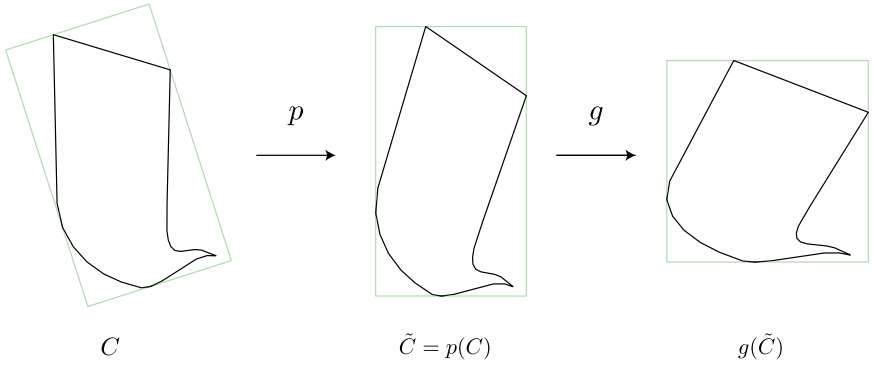


Fig. 5. Illustration of the pre-processing step. In the figure, C is the computational domain, p is the affine transformation for C in the pre-processing step, g is the initial parameterization given by the initial network. The bounding boxes are marked in green.

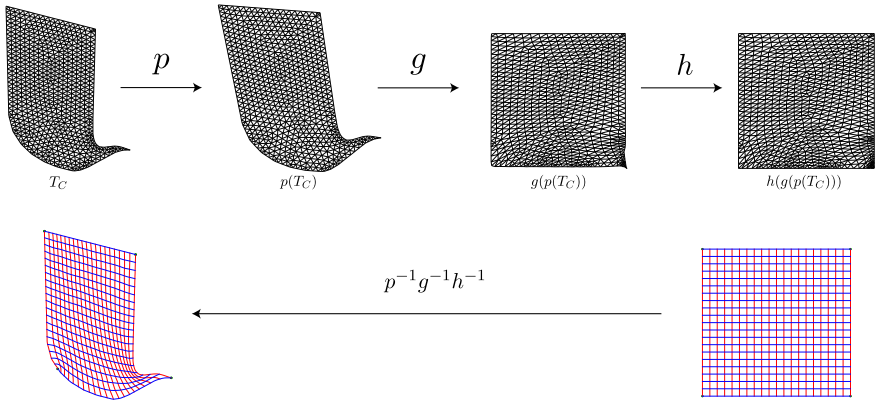


Fig. 6. Illustration of the post processing step. In the figure, T_C is the triangulation of the computational domain C, p is the affine transformation for C in the pre-processing step, g is the inverse parameterization given by our network, and h is a map that maps $g(p(T_C))$ to P.

that the map g is bijective if and only if $|\mu| < 1$. In this paper, we define the angular distortion at a point $\mathbf{p}$ by

$$L_{angle}(\mathbf{p}) = \left(\frac{\|J(\mathbf{p})\|_F^2 - 2|J(\mathbf{p})|}{\|J(\mathbf{p})\|_F^2 + 2|J(\mathbf{p})|} \right)^2, \tag{10}$$

and the angular distortion over the domain C is defined as

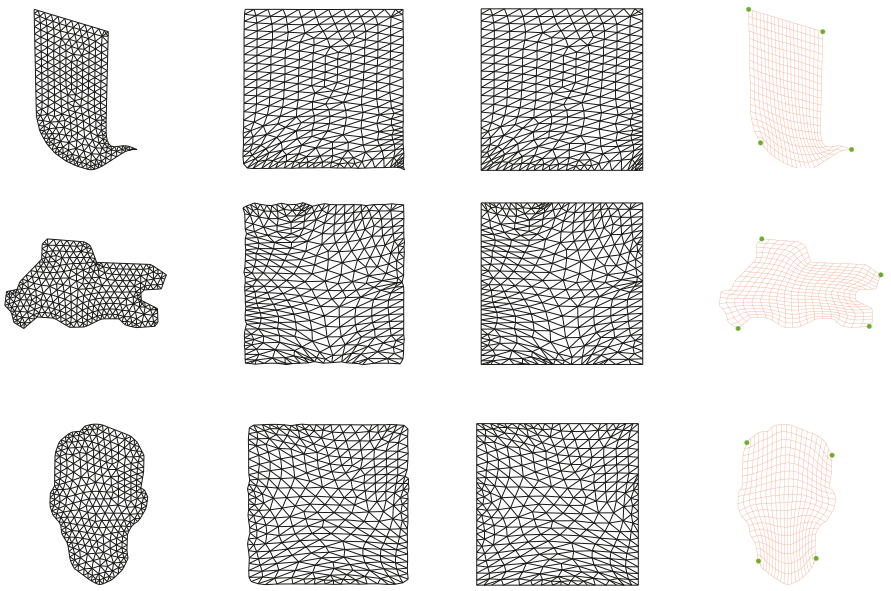
$$L_{angle}(C) = \int_C L_{angle}(\mathbf{p})\, d\mathbf{p}. \tag{11}$$

In discrete form

$$L_{angle} = \frac{1}{m} \sum_{i=1}^{m} L_{angle}(\mathbf{p}_i), \tag{12}$$

where $\{\mathbf{p}_i\}_{i=1}^m$ are a set of points uniformly sampled in C.

Boundary difference. The boundary consists of two components: L_{haus} and L_{fold}. L_{haus} is responsible for controlling the Hausdorff distance between two boundaries, while L_{fold} ensures that $O = (0.5, 0.5) \in g(C)$ and that $g(\partial C)$ has no self-intersection. The inclusion of L_{fold} aids in the



(a) The triangulated computational domain T_C. (b) $g(T_{\tilde{C}})$ (c) $h(g(T_{\tilde{C}}))$ (d) The final parameterization result.

Fig. 7. In the figure, the comparison between $g(T_{\tilde{C}})$ and $h(g(T_{\tilde{C}}))$ are shown for several computational domains.

convergence of the network. Specifically, they are defined in discrete forms as

$$L_{haus} = \frac{1}{n}\sum_{i=1}^{n} dist(g(\mathbf{q}_i), \partial P)^2 \tag{13}$$

and

$$L_{fold} = \frac{1}{n}\sum_{i=1}^{n} (s_i)_+, \tag{14}$$

where $\{\mathbf{q}_i\}_{i=1}^n$ is a set of points sampled on the boundary of C in counter-clockwise order, and s_i is the sine of the angle between vectors $g(\mathbf{q}_i) - O$ and $g(\mathbf{q}_{i+1}) - O$:

$$s_i = \frac{(g(\mathbf{q}_i) - O) \times (g(\mathbf{q}_{i+1}) - O)}{\|g(\mathbf{q}_i) - O\|_2 \|g(\mathbf{q}_{i+1}) - O\|_2}. \tag{15}$$

Here in the numerator, the cross product is just the 2×2 determinant with the two vectors as the rows. The final boundary loss is defined as: $L_{bound} = L_{haus} + w_4 L_{fold}$, where $w_4 \in \mathbf{R}$ is a positive weight, which is set to be 100 in our implementation.

3.6. Pre-processing and post processing

1. Pre-processing
   Before training the network, some affine transformations are applied to ensure compatibility between the computational domain C and the parametric domain $P = [0,1]^2$. First, principal component analysis (PCA) is utilized on C to obtain two mutually perpendicular principal directions. Subsequently, a tight oriented bounding box B is obtained to enclose C. Uniform scaling is then applied to both B and C to achieve an area of 1 for C. Afterward, an affine transformation is performed to make B axes-aligned and centered at $(0.5, 0.5)$. The above transformations are denoted as p, and $\tilde{C} = p(C)$ represents the transformed computational domain. Consequently, the network $NetDP$ can be initialized with an affine map that maps B into the unit square P.
   To achieve this, the network is initially pretrained as an identity map. As a result, the network can be adjusted to any linear transformation by modifying the parameters of its last layer. Furthermore, to satisfy the requirement $(0.5, 0.5) \in g(\tilde{C})$ stated in Lemma 2, a further translation is performed. If the point $(0.5, 0.5)$ lies outside $g(\tilde{C})$, the translation ensures that $(0.5, 0.5)$ becomes an interior point. This translation is also accomplished by modifying the last layer of the network. The pre-processing step is visualized in Fig. 5.
2. Post processing
   Let g be the inverse parameterization produced by the network $NetDP$. In general, the mapped boundary $g(\tilde{C})$ of the computational domain $\tilde{C}$ may not align well with the boundary of the parametric domain P. To address this issue, a post-processing step is necessary. Initially, we generate a triangulation T_C for the computational domain C, and then obtain $T_{\tilde{C}} = p(T_C)$ as a triangulation of $\tilde{C} = p(C)$. Hence, $g(T_{\tilde{C}}) = g(p(T_C))$ represents a triangulation of P, but the boundary is not well-fitted. To achieve a better fit between $g(T_{\tilde{C}})$ and the boundary of P, we modify $g(T_{\tilde{C}})$ such that $g(T_{\tilde{C}})$ fits the boundary of P well. To do so, we employ the mean value coordinates (MVC) of the inner points of $g(T_{\tilde{C}})$ (please refer to [72] for details). After that, we can preserve the MVC of the inner points while moving the boundary points to nearby places on ∂P. Essentially, we find a bijective map h such that $h(g(T_{\tilde{C}})) = P$, and its bijectivity is guaranteed by [73]. In this way, $h \circ g(\tilde{C})$ fits P with improved accuracy.
   Finally, we need to compute a tensor product B-spline f to approximate $(h \circ g \circ p)^{-1}$, where p represents the affine transformation computed in the pre-processing step. This approximation can be accomplished by solving the following least square problem:

$$\min \sum_{\mathbf{v} \in V(T_C)} \|f((h \circ g \circ p)(\mathbf{v})) - \mathbf{v}\|_2^2, \tag{16}$$

here $V(T_C)$ is the vertex set of T_C. Fig. 6 illustrates the whole process and Fig. 7 shows the effects of post-processing.

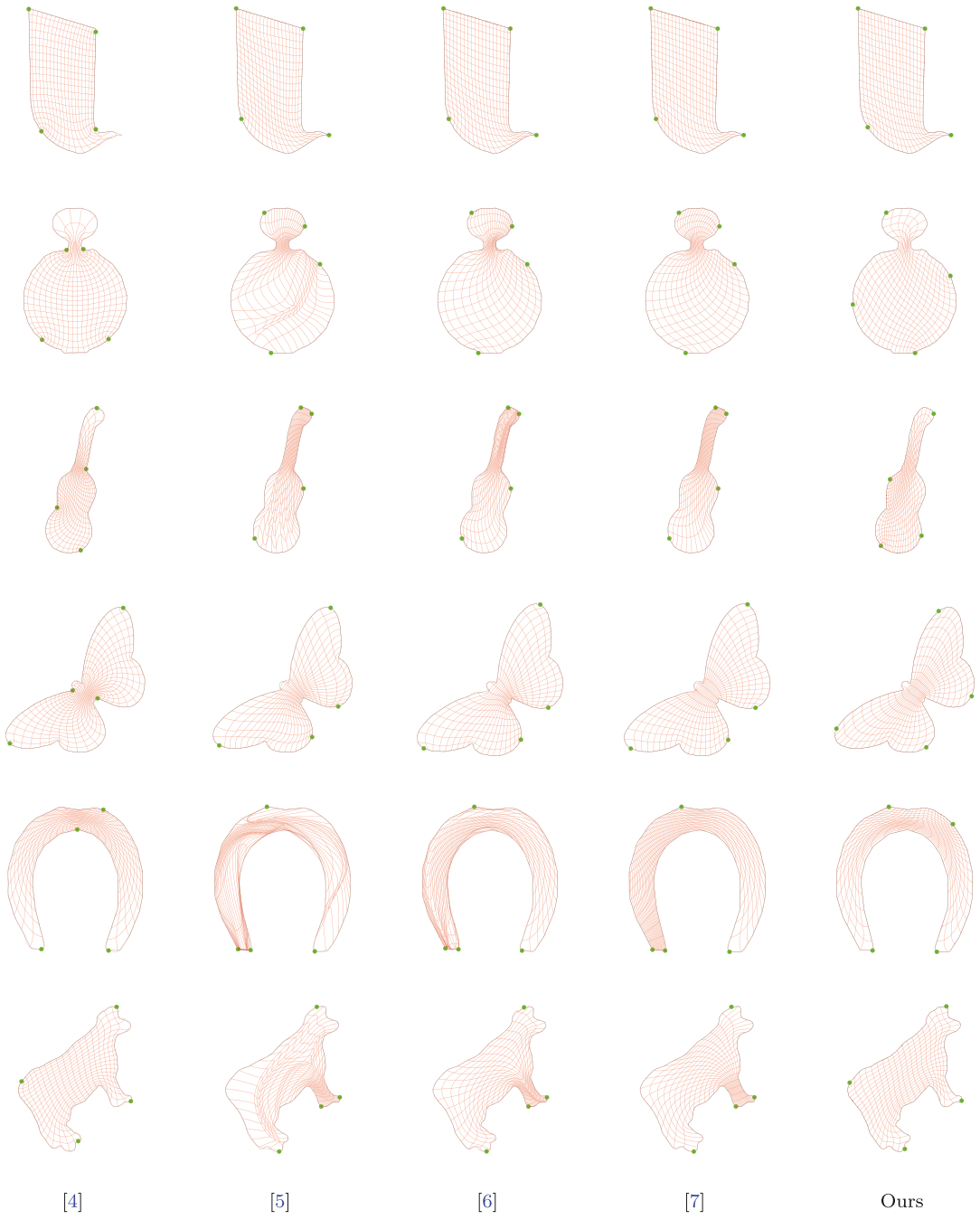


Fig. 8. Parameterization results given by our method and the methods in [4–7] on some computational domains in the MPEG-7 dataset. The boundaries of the computational domains are marked in gray.

Table 1
The distortion and bijectivity of the parameterization results given by our method and the method in [4–7] on MPEG-7 dataset. For methods in [5–7], we apply the method in [2] to generate proper boundary correspondence. The quantity in the table is the average of all the models in the dataset.

	$Area_{ave}$	$Area_{max}$	$Angle_{ave}$	$Angle_{max}$	$Bijectivity$	$Bound_{dis}$	$Time$ (s)
[4]	0.14534	0.99211	**0.16350**	0.97309	109/977	0.01364	27.6
[5]+[2]	0.27634	0.88470	0.30951	0.69960	854/977	0.01130	637.1
[6]+[2]	0.22444	0.91145	0.27246	0.78120	772/977	0.00917	24.1
[7]+[2]	0.34081	0.83906	0.36248	0.78276	737/977	0.00917	**0.2**
Our method	**0.11043**	**0.61166**	0.21822	**0.65673**	**955**/977	**0.00483**	17.3

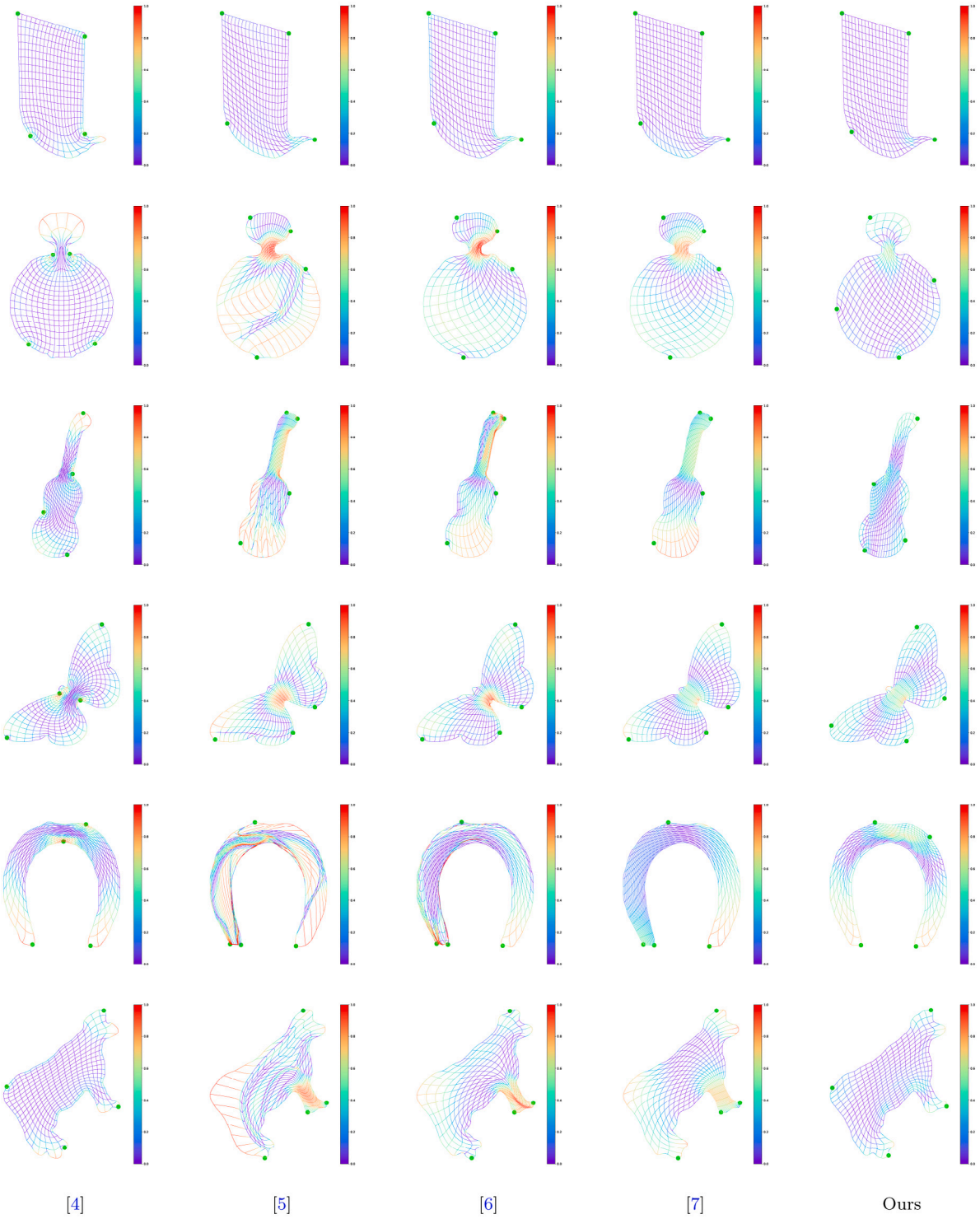


Fig. 9. Parameterization results given by our method and the methods in [4–7] on some computational domains in the MPEG-7 dataset. The area distortion is colored on each domain.

4. Evaluation

We implement our method and conduct experiments using the MPEG-7 dataset, which includes 977 2D domains described by polygonal boundaries. The code is implemented in python, using the PyTorch framework and cuda acceleration. The experimental results demonstrate the effectiveness of our approach in computing bijective and low-distortion parameterization for planar domains. Compared with the state-of-the-art parameterization method in previous work [4–8], our approach achieves superior boundary correspondence, lower parameterization distortion and much higher likelihood of bijectivity.

The training process of the network $NetDP$ is performed in an unsupervised manner, utilizing the loss function defined in (1). The training process consists of two steps: (1) In the first 1500 iterations, we assign weights $w_1 = 10^{-1}$, $w_2 = 10^{-1}$ and $w_3 = 10$. The relatively small weight w_3 leaves room for shape deformation. (2) In the second phase, we adjust the weights as follows: when $L_{haus} > 10^{-5}$, we set $w_1 = 10^{-3}$, $w_2 = 3 \times 10^{-2}$ and $w_3 = 100$. The large weight w_3 leads to

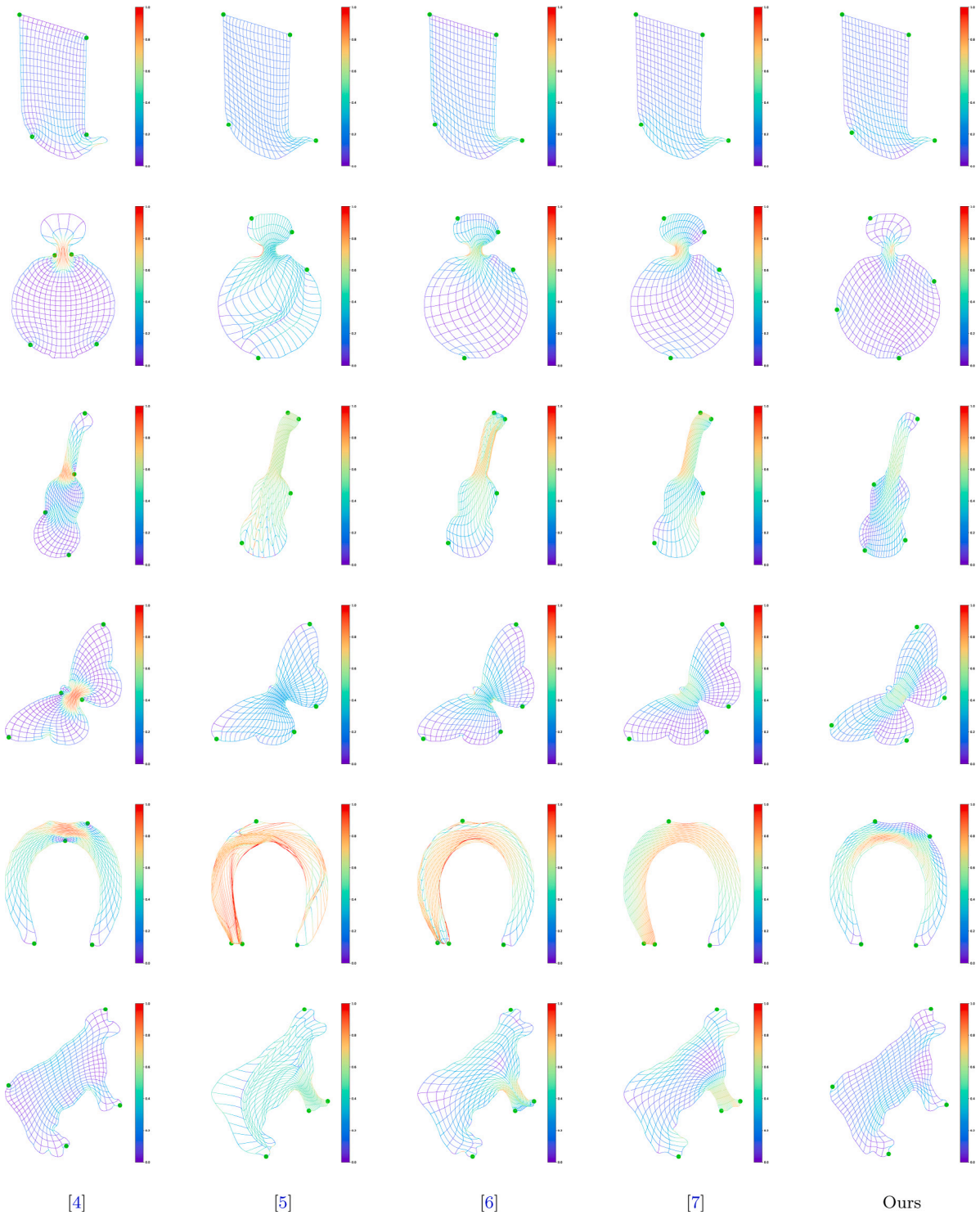


Fig. 10. Parameterization results given by our method and the methods in [4–7] on some computational domains in the MPEG-7 dataset. The angular distortion is colored on each domain.

fast convergence of the boundary map. However, when $L_{haus} \leq 10^{-5}$, we set $w_1 = 10^{-3}$, $w_2 = 3 \times 10^{-2}$ and $w_3 = 1$. The second phase of training continues until the network converges or reaches the maximum number of iterations.

To evaluate the performance of our method, we conduct parameterization experiments on the entire MPEG-7 dataset and compare the results with those obtained by state-of-the-art methods in [4–7]. The methods in [5–7] are traditional approaches that require given boundary correspondences as inputs. To generate suitable boundary correspondences for these methods, we utilize the method presented in [2]. We evaluate the results using 7 metrics: $Area_{ave}$, $Area_{max}$, $Angle_{ave}$, $Angle_{max}$, $Bound_{dis}$, $Bjiectivity$ and $Time$. These metrics represent the average area distortion, maximal area distortion, average angular distortion, maximal angular distortion, the Hausdorff distance between two boundaries $g(\partial P)$ and ∂C, bijectivity and the computational time, respectively. Here the area distortion and angular distortion

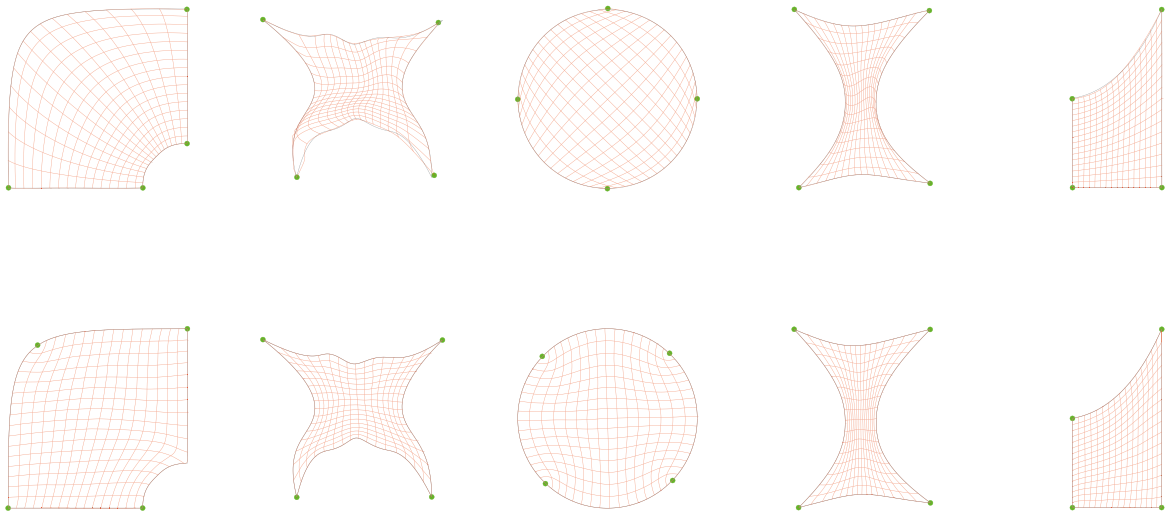


Fig. 11. Comparisons between our approach and [8]. The top row shows the result by [8] while the pictures in the second row are our results. The boundaries of the computational domains are marked in gray.

are defined as

$$\begin{aligned} Area &= 1 - \frac{1}{(|J|_+ + 1/|J|_+ - 1)}, \\ Angle &= \min\left(\frac{\|J\|_F^2 - 2|J|}{\|J\|_F^2 + 2|J|}, 1\right) \end{aligned} \tag{17}$$

Both area distortion and angular distortion have a range of $[0, 1]$, and a smaller value indicates less distortion. We compute these metrics for all the models in the MPEG-7 dataset and present the average values in Table 1.

As shown in Table 1, our method outperforms the methods [4–7] in terms of parameterization quality, except for average angular distortion. Notably, our networks achieve a bijective parameterization rate of 98% for the models in the dataset, while the method by [4] only achieves a bijective rate of $1/9$. As for the computational cost, the method by [7] is the most efficient one and our method ranks second.

Fig. 8 shows some parameterization results qualitatively, while the corresponding area and angular distortions for each domain are illustrated in Figs. 9 and 10. From these figures, several advantages of our method over the methods in [4–7] can be observed. First, our method achieves higher accuracy in boundary maps. As shown in Fig. 8, the boundaries obtained by the parametric maps generated by [4–7] often failed to align with the target boundaries. Second, our method provides better correspondence of corner points. The corner points produced by [4] often lie on concave regions, resulting in non-bijective parameterizations. Furthermore, the corner points selected by [2] are occasionally unevenly distributed. Thirdly, our method yields more uniform parameterization results compared to the other methods, especially for complex models. Finally, large distortions often occur in certain regions of the computational domains by the other methods, while our method avoids this issue.

Finally, we conduct some comparisons with the method [8], a neural network based approach employing PINN. Due to its incapability to handle complex non-convex domains, we were unable to perform comparisons on the MPEG-7 dataset. Instead, we conducted comparative experiments on several simpler domains tested in [8]. The results are presented in Fig. 11 and Table 2. From these results, it can be seen that our method achieves higher accuracy in boundary maps and yields smaller area and angle distortion except for example 5. Notice that the boundary correspondences in the first and the third examples are different, which in fact reflects the advantage of our method—it optimizes not only the interior domain but also the boundary correspondence.

At last, we would like to remark that our method occasionally generates non-bijective results. Fig. 12 illustrates some failure examples. It can be observed from the figures that computational domains with complex boundaries may pose challenges for our method. There are two primary reasons for these difficulties. First, the complex boundary makes the internal optimization challenging, resulting in suboptimal local properties of the parameterized results, as evident in Figs. 12(a) and 12(b). Secondly, the complexity of the parameterization can lead to non-bijectivity in the spline fitting step during post-processing, as shown in Figs. 12(c)–12(e). However, we have found only 22 failure cases among 977 examples in the dataset.

5. Conclusion

In this paper, we present a novel approach for simultaneous boundary and interior parameterization of planar domains using deep learning techniques. An unsupervised neural network $NetDP$ is constructed to approximate the inverse parameterization. Following the computation of the inverse parameterization, we employ a tensor product B-spline to fit the forward map. Compared to the state-of-the-art algorithms [4–7], our method achieves higher bijectivity ratio, lower distortion and more uniform parameterization. Experimental examples are provided to illustrate the effectiveness of our approach.

However, there are a few limitations in the current approach. First, for each computational domain, the network $NetDP$ needs to be re-trained from scratch, which affects the computational efficiency. Secondly, our method is designed for two-dimensional cases and cannot be directly applied to three-dimensional tasks. Thirdly, our method can only parameterize simply connected domains by single-patch parameterization. Thus, an important future research direction is to develop a network which can immediately produce high quality parameterization without training once it has been trained. It is also interesting to generalize the current work and extend it to the parameterization of three-dimensional computational volumes. Finally, developing a network which divides multi-genus geometry into multiple genus-zero geometries, and then parameterize the domain by parts is another important research problem worthy of future investigation.

Table 2
Comparisons between our approach and [8] among 5 domains shown in Fig. 11.

Method	Domain	$Area_{ave}$	$Area_{max}$	$Angle_{ave}$	$Angle_{max}$	$Bound_{Haus}$	*Bijective*
[8]	1	0.20443	0.81641	0.05058	0.52985	0.00948	Yes
Our method	1	**0.00517**	**0.34448**	**0.03330**	**0.41521**	**0.00534**	Yes
[8]	2	0.22373	1.00000	0.15638	1.00000	0.28332	No
Our method	2	**0.04841**	**0.51279**	**0.12230**	**0.64688**	**0.00460**	Yes
[8]	3	0.09699	0.90961	0.07023	0.81400	0.01126	Yes
Our method	3	**0.00633**	**0.32495**	**0.04022**	**0.32502**	**0.00321**	Yes
[8]	4	0.17989	0.90780	0.30994	0.89120	0.04265	Yes
Our method	4	**0.14398**	**0.56426**	**0.29318**	**0.58502**	**0.00297**	Yes
[8]	5	0.03763	**0.14027**	**0.08651**	**0.37500**	0.01518	Yes
Our method	5	**0.01654**	0.17094	0.09337	0.43537	**0.00111**	Yes

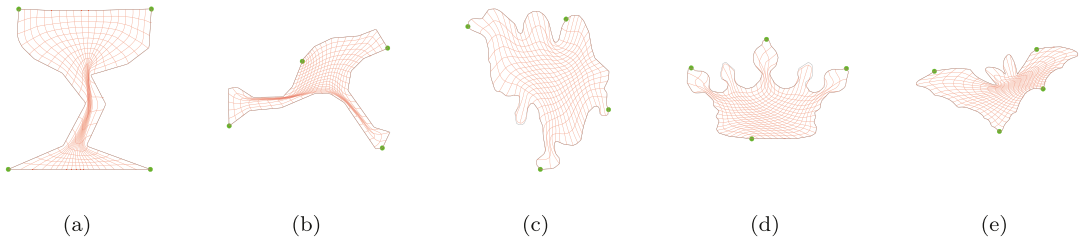


Fig. 12. Some non-bijective results of our method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to thank Dr. Ye Zheng, Maodong Pan, Xianshun Nian, Ye Ji and Antonella Falini for providing partial experimental results and the source codes. This work is supported by NSF of China (no.61972368, no.12371383).

References

[1] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Engrg 2005;194(39):4135–95. http://dx.doi.org/10.1016/j.cma.2004.10.008.

[2] Zheng Y, Pan M, Chen F. Boundary correspondence of planar domains for isogeometric analysis based on optimal mass transport. Comput Aided Des 2019;114:28–36. http://dx.doi.org/10.1016/j.cad.2019.04.008.

[3] Zheng Y, Chen F. Volumetric boundary correspondence for isogeometric analysis based on unbalanced optimal transport. Comput Aided Des 2021;140:103078. http://dx.doi.org/10.1016/j.cad.2021.103078.

[4] Liu H, Yang Y, Liu Y, Fu X-M. Simultaneous interior and boundary optimization of volumetric domain parameterizations for IGA. Comput Aided Geom Design 2020;79:101853. http://dx.doi.org/10.1016/j.cagd.2020.101853.

[5] Nian X, Chen F. Planar domain parameterization for isogeometric analysis based on Teichmüller mapping. Comput Methods Appl Mech Engrg 2016;311:41–55. http://dx.doi.org/10.1016/j.cma.2016.07.035.

[6] Pan M, Chen F, Tong W. Low-rank parameterization of planar domains for isogeometric analysis. Comput Aided Geom Design 2018;63:1–16. http://dx.doi.org/10.1016/j.cagd.2018.04.002.

[7] Ji Y, Chen K, Möller M, Vuik C. On an improved PDE-based elliptic parameterization method for isogeometric analysis using preconditioned Anderson acceleration. Comput Aided Geom Design 2023;102:102191. http://dx.doi.org/10.1016/j.cagd.2023.102191.

[8] Falini A, D'Inverno GA, Sampoli ML, Mazzia F. Splines parameterization of planar domains by physics-informed neural networks. Mathematics 2023;11(10). http://dx.doi.org/10.3390/math11102406, URL: https://www.mdpi.com/2227-7390/11/10/2406.

[9] Gravesen J, Evgrafov A, Nguyen M, Nørtoft P. Planar parametrization in isogeometric analysis. In: Mathematical methods for curves and surfaces. Springer Berlin Heidelberg; 2014, p. 189–212.

[10] Martin T, Cohen E, Kirby R. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Comput Aided Geom Des 2009;26(6):648–64. http://dx.doi.org/10.1016/j.cagd.2008.09.008.

[11] Nguyen T, Jüttler B. Parameterization of contractible domains using sequences of harmonic maps. In: Curves and surfaces. Lecture notes in computer science, Springer, Berlin, Heidelberg; 2010, p. 501–14. http://dx.doi.org/10.1007/978-3-642-27413-8_32.

[12] Xu G, Mourrain B, Duvigneau R, Galligo A. Constructing analysis-suitable parameterization of computational domain from CAD boundary by variational harmonic method. J Comput Phys 2013;252:275–89. http://dx.doi.org/10.1016/j.jcp.2013.06.029.

[13] Xu G, Mourrain B, Duvigneau R, Galligo A. Variational harmonic method for parameterization of computational domain in 2D isogeometric analysis. In: 2011 12th international conference on computer-aided design and computer graphics. 2011, p. 223–8. http://dx.doi.org/10.1109/CAD/Graphics.2011.22.

[14] Xu G, Mourrain B, Duvigneau R, Galligo A. Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications. Comput-Aided Des 2013;45(2):395–404. http://dx.doi.org/10.1016/j.cad.2012.10.022.

[15] Wang X, Qian X. An optimization approach for constructing trivariate B-spline solids. Comput-Aided Des 2014;46:179–91. http://dx.doi.org/10.1016/j.cad.2013.08.030.

[16] Pan M, Chen F, Tong W. Volumetric spline parameterization for isogeometric analysis. Comput Methods Appl Mech Eng 2020;359. http://dx.doi.org/10.1016/j.cma.2019.112769.

[17] Escobar J, Cascón J, Rodríguez E, Montenegro R. A new approach to solid modeling with trivariate T-splines based on mesh optimization. Comput Methods Appl Mech Engrg 2011;200(45):3210–22. http://dx.doi.org/10.1016/j.cma.2011.07.004, URL: https://www.sciencedirect.com/science/article/pii/S0045782511002386.

[18] Zhang Y, Wang W, Hughes TJ. Solid T-spline construction from boundary representations for genus-zero geometry. Comput Methods Appl Mech Engrg 2012;249–252:185–97. http://dx.doi.org/10.1016/j.cma.2012.01.014, URL: https://www.sciencedirect.com/science/article/pii/S0045782512000254, Higher Order Finite Element and Isogeometric Methods.

[19] Wang W, Zhang Y, Liu L, Hughes TJ. Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology. Comput Aided Des 2013;45(2):351–60. http://dx.doi.org/10.1016/j.cad.2012.10.018, Solid and Physical Modeling 2012.

[20] Falini A, Špeh J, Jüttler B. Planar domain parameterization with THB-splines. Comput Aided Geom Design 2015;35–36:95–108. http://dx.doi.org/10.1016/j.cagd.2015.03.014, Geometric Modeling and Processing 2015.

[21] Chan C, Anitescu C, Rabczuk T. Volumetric parametrization from a level set boundary representation with PHT-splines. Comput-Aided Des 2017;82:29–41. http://dx.doi.org/10.1016/j.cad.2016.08.008.

[22] Pan M, Chen F. Constructing planar domain parameterization with HB-splines via quasi-conformal mapping. Comput Aided Geom Design 2022;97:102133. http://dx.doi.org/10.1016/j.cagd.2022.102133, URL: https://www.sciencedirect.com/science/article/pii/S0167839622000693.

[23] Zheng Y, Chen F. Volumetric parameterization with truncated hierarchical B-splines for isogeometric analysis. Comput Methods Appl Mech Engrg 2022;401:115662. http://dx.doi.org/10.1016/j.cma.2022.115662, URL: https://www.sciencedirect.com/science/article/pii/S004578252200617X.

[24] Pauley M, Nguyen D-M, Mayer D, Špeh J, Weeger O, Jüttler B. The isogeometric segmentation pipeline. In: Jüttler B, Simeon B, editors. Isogeometric analysis and applications 2014. Lecture notes in computational science and engineering, Cham: Springer International Publishing; 2015, p. 51–72. http://dx.doi.org/10.1007/978-3-319-23315-4_3.

[25] Xu J, Chen F, Deng J. Two-dimensional domain decomposition based on skeleton computation for parameterization and isogeometric analysis. Comput Methods Appl Mech Eng 2015;284(SI):541–55. http://dx.doi.org/10.1016/j.cma.2014.09.026.

[26] Buchegger F, Jüttler B. Planar multi-patch domain parameterization via patch adjacency graphs. Comput Aided Des 2017;82:2–12. http://dx.doi.org/10.1016/j.cad.2016.05.019.

[27] Xu G, Li M, Mourrain B, Rabczuk T, Xu J, Bordas SP. Constructing IGA-suitable planar parameterization from complex CAD boundary by domain partition and global/local optimization. Comput Methods Appl Mech Engrg 2018;328:175–200. http://dx.doi.org/10.1016/j.cma.2017.08.052.

[28] Xiao S, Kang H, Fu X-M, Chen F. Computing IGA-suitable planar parameterizations by PolySquare-enhanced domain partition. Comput Aided Geom Design 2018;62:29–43. http://dx.doi.org/10.1016/j.cagd.2018.03.008.

[29] Chen L, Xu G, Wang S, Shi Z, Huang J. Constructing volumetric parameterization based on directed graph simplification of ℓ1 polycube structure from complex shapes. Comput Methods Appl Mech Engrg 2019;351:422–40. http://dx.doi.org/10.1016/j.cma.2019.01.036.

[30] Hu K, Zhang YJ. Centroidal Voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. Comput Methods Appl Mech Engrg 2016;305:405–21. http://dx.doi.org/10.1016/j.cma.2016.03.021.

[31] Liu L, Zhang Y, Liu Y, Wang W. Feature-preserving T-mesh construction using skeleton-based polycubes. Comput Aided Des 2015;58:162–72. http://dx.doi.org/10.1016/j.cad.2014.08.020.

[32] Hu K, Zhang YJ, Liao T. Surface segmentation for polycube construction based on generalized centroidal Voronoi tessellation. Comput Methods Appl Mech Engrg 2017;316:280–96. http://dx.doi.org/10.1016/j.cma.2016.07.005.

[33] Liu L, Zhang Y, Hughes T, Scott M, Sederberg T. Volumetric T-spline construction using Boolean operations. Eng Comput 2014;30(4):425–39. http://dx.doi.org/10.1007/s00366-013-0346-6.

[34] Zhang Y, Bazilevs Y, Goswami S, Bajaj CL, Hughes TJR. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Comput Methods Appl Mech Engrg 2007;196(29):2943–59. http://dx.doi.org/10.1016/j.cma.2007.02.009.

[35] Wang S, Ren J, Fang X, Lin H, Xu G, Bao H, et al. IGA-suitable planar parameterization with patch structure simplification of closed-form polysquare. Comput Methods Appl Mech Engrg 2022;392:114678. http://dx.doi.org/10.1016/j.cma.2022.114678, URL: https://www.sciencedirect.com/science/article/pii/S0045782522000652.

[36] Blidia A, Mourrain B, Xu G. Geometrically smooth spline bases for data fitting and simulation. Comput Aided Geometric Des 2020;78. http://dx.doi.org/10.1016/j.cagd.2020.101814.

[37] Kapl M, Sangalli G, Takacs T. Construction of analysis-suitable G(1) planar multi-patch parameterizations. Comput-Aided Des 2018;97:41–55. http://dx.doi.org/10.1016/j.cad.2017.12.002.

[38] Kapl M, Sangalli G, Takacs T. Dimension and basis construction for analysis-suitable G(1) two-patch parameterizations. Comput Aided Geometric Des 2017;52–53:75–89. http://dx.doi.org/10.1016/j.cagd.2017.02.013.

[39] Kapl M, Buchegger F, Bercovier M, Juttler B. Isogeometric analysis with geometrically continuous functions on planar multi-patch geometries. Comput Methods Appl Mech Eng 2017;316:209–34. http://dx.doi.org/10.1016/j.cma.2016.06.002.

[40] Kapl M, Sangalli G, Takacs T. An isogeometric C-1 subspace on unstructured multi-patch planar domains. Comput Aided Geometric Des 2019;69:55–75. http://dx.doi.org/10.1016/j.cagd.2019.01.002.

[41] Speleers H, Manni C. Optimizing domain parameterization in isogeometric analysis based on Powell–Sabin splines. J Comput Appl Math 2015;289:68–86. http://dx.doi.org/10.1016/j.cam.2015.03.024.

[42] Xie J, Xu J, Dong Z, Xu G, Deng C, Mourrain B, et al. Interpolatory catmull-clark volumetric subdivision over unstructured hexahedral meshes for modeling and simulation applications. Comput Aided Geom Design 2020;80:101867. http://dx.doi.org/10.1016/j.cagd.2020.101867, URL: https://www.sciencedirect.com/science/article/pii/S0167839620300546.

[43] Xu G, Kwok T-H, Wang CC. Isogeometric computation reuse method for complex objects with topology-consistent volumetric parameterization. Comput Aided Des 2017;91:1–13. http://dx.doi.org/10.1016/j.cad.2017.04.002, URL: https://www.sciencedirect.com/science/article/pii/S0010448517300477.

[44] Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, et al. Deep learning for generic object detection: A survey. Int J Comput Vis 2020;128. http://dx.doi.org/10.1007/s11263-019-01247-4.

[45] Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: A survey. IEEE Trans Pattern Anal Mach Intell 2021;1. http://dx.doi.org/10.1109/TPAMI.2021.3059968.

[46] Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J. Deep learning–based text classification: A comprehensive review. ACM Comput Surv 2021;54(3). http://dx.doi.org/10.1145/3439726.

[47] Wang D, Xu J, Gao F, Wang CC, Gu R, Lin F, et al. IGA-reuse-NET: A deep-learning-based isogeometric analysis-reuse approach with topology-consistent parameterizationimage 1. Comput Aided Geom Design 2022;95:102087. http://dx.doi.org/10.1016/j.cagd.2022.102087, URL: https://www.sciencedirect.com/science/article/pii/S0167839622000231.

[48] Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, et al. A survey on deep learning in medical image analysis. Med Image Anal 2017;42:60–88. http://dx.doi.org/10.1016/j.media.2017.07.005.

[49] Suganyadevi S, Seethalakshmi V, Balasamy K. A review on deep learning in medical image analysis. Int J Multimed Inf Retriev 2022;11(1):19–38. http://dx.doi.org/10.1007/s13735-021-00218-1.

[50] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th international conference on neural information processing systems. Curran Associates Inc.; 2016, p. 3844–52.

[51] Monti F, Boscaini D, Masci J, Rodolà E, Svoboda J, Bronstein MM. Geometric deep learning on graphs and manifolds using mixture model CNNs. In: 2017 IEEE conference on computer vision and pattern recognition. 2017, p. 5425–34. http://dx.doi.org/10.1109/CVPR.2017.576.

[52] Gong S, Chen L, Bronstein M, Zafeiriou S. SpiralNet++: A fast and highly efficient mesh convolution operator. In: 2019 IEEE/CVF international conference on computer vision workshop. 2019, p. 4141–8. http://dx.doi.org/10.1109/ICCVW.2019.00509.

[53] Fey M, Lenssen JE, Weichert F, Müller H. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. 2018, p. 869–77. http://dx.doi.org/10.1109/CVPR.2018.00097.

[54] Litany O, Remez T, Rodolà E, Bronstein A, Bronstein M. Deep functional maps: Structured prediction for dense shape correspondence. In: 2017 IEEE international conference on computer vision. 2017, p. 5660–8. http://dx.doi.org/10.1109/ICCV.2017.603.

[55] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: A network with an edge. ACM Trans Graph 2019;38(4):90:1–90:12. http://dx.doi.org/10.1145/3306346.3322959.

[56] Qi CR, Yi L, Su H, Guibas LJ. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems, vol. 30. Curran Associates, Inc.; 2017, p. 5099–108.

[57] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. ACM Trans Graph 2019;38(5):146:1–146:12. http://dx.doi.org/10.1145/3326362.

[58] Dai A, Ruizhongtai Qi C, Nießner M. Shape completion using 3D-encoder-predictor CNNS and shape synthesis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 5868–77.

[59] Bouritsas G, Bokhnyak S, Ploumpis S, Bronstein M, Zafeiriou S. Neural 3D morphable models: Spiral convolutional networks for 3D shape representation learning and generation. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 7213–22.

[60] Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. 2017, arXiv:1710.10903.

[61] Wen C, Zhang Y, Li Z, Fu Y. Pixel2Mesh++: Multi-view 3D mesh generation via deformation. In: 2019 IEEE/CVF international conference on computer vision. 2019, p. 1042–51. http://dx.doi.org/10.1109/ICCV.2019.00113.

[62] Gao Z, Yan J, Zhai G, Zhang J, Yang Y, Yang X. Learning local neighboring structure for robust 3D shape representation. 2020, arXiv:2004.09995.

[63] Scholz F, Jüttler B. Parameterization for polynomial curve approximation via residual deep neural networks. Comput Aided Geom Design 2021;85:101977. http://dx.doi.org/10.1016/j.cagd.2021.101977.

[64] Wang H, Zhang J. A survey of deep Learning-Based mesh processing. Commun Math Stat 2022;10(1):163–94.

[65] Cao W, Yan Z, He Z, He Z. A comprehensive survey on geometric deep learning. IEEE Access 2020;8:35929–49. http://dx.doi.org/10.1109/ACCESS.2020.2975067.

[66] Xiao Y-P, Lai Y-K, Zhang F-L, Li C, Gao L. A survey on deep geometry learning: From a representation perspective. Comput Vis Media 2020;6(2):113–33. http://dx.doi.org/10.1007/s41095-020-0174-8.

[67] Giannelli C, Imperatore S, Mantzaflaris A, Scholz F. Learning meshless parameterization with graph convolutional neural networks. Neural Comput Appl 2023.

[68] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.

[69] Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. NeRF: Representing scenes as neural radiance fields for view synthesis. 2020, arXiv:2003.08934.
[70] Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 165–74.
[71] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst 2017;30.
[72] Floater MS. Mean value coordinates. Comput Aided Geom Design 2003;20(1):19–27. http://dx.doi.org/10.1016/S0167-8396(03)00002-5, URL: https://www.sciencedirect.com/science/article/pii/S0167839603000025.
[73] Tutte WT. How to draw a graph. Proc Lond Math Soc 1963;13:743–67. http://dx.doi.org/10.1112/plms/s3-13.1.743.