Communications in Mathematics and Statistics (2023) 11:131–150
https://doi.org/10.1007/s40304-023-00337-7

Boundary Correspondence for Iso-Geometric Analysis Based on Deep Learning

Zheng Zhan[1] · Ye Zheng[1] · Wenping Wang[2] · Falai Chen[1]

Received: 8 February 2022 / Revised: 29 August 2022 / Accepted: 2 February 2023 /
Published online: 16 March 2023

Abstract
One of the key problems in isogeometric analysis(IGA) is domain parameterization, i.e., constructing a map between a parametric domain and a computational domain. As a preliminary step of domain parameterization, the mapping between the boundaries of the parametric domain and the computational domain should be established. The boundary correspondence strongly affects the quality of domain parameterization and thus subsequent numerical analysis. Currently, boundary correspondence is generally determined manually and only one approach based on optimal mass transport discusses automatic generation of boundary correspondence. In this article, we propose a deep neural network based approach to generate boundary correspondence for 2D simply connected computational domains. Given the boundary polygon of a planar computational domain, the main problem is to pick four corner vertices on the input boundary in order to subdivide the boundary into four segments which correspond to the four sides of the parametric domain. We synthesize a dataset with corner correspondence and train a fully convolutional network to predict the likelihood of each boundary vertex to be one of the corner vertices, and thus to locate four corner vertices with locally maximum likelihood. We evaluate our method on two types of datasets: MPEG-7 dataset and CAD model dataset. The experiment results demonstrate that

✉ Falai Chen
chenfl@ustc.edu.cn

Zheng Zhan
zz00822@mail.ustc.edu.cn

Ye Zheng
zhengye@mail.ustc.edu.cn

Wenping Wang
wenping@cs.hku.hk

1 School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, Anhui, People's Republic of China

2 College of Architecture, Texas A&M University, College Station, TX 77843-3112, USA

our algorithm is faster by several orders of magnitude, and at the same time achieves smaller average angular distortion, more uniform area distortion and higher success rate, compared to the traditional optimization-based method. Furthermore, our neural network exhibits good generalization ability on new datasets.

Keywords Isogeometric analysis · Domain parameterization · Boundary correspondence · Deep neural network

Mathematics Subject Classification 68T07 · 68U05 · 68U07 · 65D17

1 Introduction

It is a fundamental task in isogeometric analysis (IGA) [19] to parameterize a given physical domain using splines, which is known as domain parameterization. The goal of domain parametrization is to construct a map from a conventional parametric domain to the computational domain, as illustrated in Fig. 1 for planar case. Before setting up the mapping of the interior of the domain, a correspondence between the boundaries of the parametric domian and the physical domain is required. The quality of the boundary correspondence affects the quality of domain parameterization and subsequent analysis [6, 40]. Currently, boundary correspondence is usually provided manually, and only very few optimization-based methods have been proposed to solve this problem automatically[61, 62].

In this paper, we propose a method based on deep neural network to solve the boundary correspondence problem for planar domain parameterization. This problem is reduced to find four vertices on the input boundary that correspond to the four corner points of the parametric domain–the unit square, as illustrated in Fig. 1. We tackle this problem by constructing a convolutional network which takes the discrete representation (a simple closed polygon) of the boundary as the input and gives the likelihood of each vertex of the input polygon to be a corner vertex as the output. The vertices with local maximum likelihood are then chosen as corner vertices. Since there is no real-world dataset publically available for this problem, we construct a dataset from MPEG-7, and four corner vertices for each domain boundary are generated by selecting the best set of vertices among randomly generated sets of four tuple vertices (the criteria of best is based on the quality of the subsequent parametrization using the parameterization method proposed in [38]), and by the results of the traditional optimization-based method [62].

To evaluate our method, We compare our neural network with the method in [62] on two datasets: MPEG-7 dataset and CAD dataset constructed by projecting a set of three-dimensional CAD models onto two-dimensional planes. We apply cross-validation to ensure the reliability of the results on the MPEG-7 dataset. Experiments show that our method is faster by several orders of magnitude than the method in [62] and results in domain parameterizations with smaller average angular distortion, more uniform area distortion and higher success (bijectivity) rate. In addition, our neural network exhibits good generalization ability on the CAD model dataset though it is trained on the dataset MPEG-7.

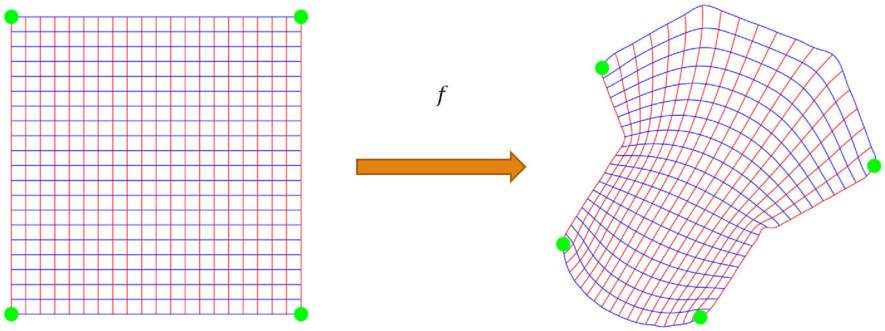


Fig. 1 A map from a parametric domain to a computational domain

The remainder of the paper is organized as follows. In Sect. 2, we review some related works on domain parameterization, boundary correspondence and geometric deep learning. Then in Sect. 3, we describe our method in details, including the architecture of our network, the loss function and the construction of the training data. In Sect. 4, we compare our neural network with the previous method on boundary correspondence construction [62] and present some experimental results. Finally, in Sect. 5 we conclude our work and list some future research problems.

2 Related Work

2.1 Domain Parameterization

Constructing a spline surface from given boundary curves is a fundamental problem in computer aided geometric design (CAGD). Coons patches [9] are a classical and popular solution to this problem. In recent years, this problem has regained much attention for its application in IGA known as domain parameterization, where a bijective map from a parametric domain to a physical domain is required. Given the boundary curve of a simply connected domain, simple and computational inexpensive methods like Coons patches or the spring model [13] usually generate non-bijective results. To overcome this challenge, various methods have been proposed in the past decade. These include harmonic maps [36, 55], quadratic programming [54, 59], Teichmüller mapping and quasi-conformal mapping [37, 38], low-rank parameterization [32, 38], and others [16, 20, 29, 56]. For complex domains with high genus, multi-patch parameterization is usually employed to tackle the problem [4, 5, 17, 18, 27, 28, 39, 48, 49, 53, 57, 58, 60]. However, the global smoothness of multi-patch parameterization is a tough problem, especially in 3D case [2, 21–24, 45].

2.2 Boundary Correspondence

Setting up a meaningful correspondence between two shapes is one of the essential tasks in shape analysis, which usually requires an understanding of the local or global

structure of the shapes in a specific problem. Various methods [43, 46] have been developed under different assumptions about the shapes in question, for example, deformation between shapes is sometimes assumed to be nearly-isometric, or shapes are different postures of a same body. For our problem, unfortunately, these methods do not work well since the assumptions they rely on do not hold. Based on curvature similarity between the parametric domain and the physical domain, Zheng et al. [61, 62] proposed automatic methods for planar and volumetric boundary correspondence based on optimal mass transport theory.

2.3 Geometric Deep Learning

In recent years, neural networks have been widely used in computer vision [30, 33], numerical PDEs [51], natural language processing [34], health care [26] and many other fields Deep learning has also been applied in the field of geometric processing to tackle various problems such as shape classification[8, 12, 35], shape correspondence[10, 12, 25], shape segmentation[14, 41, 50], shape completion[7, 25], shape generation [3, 11, 35, 47, 52] and point data parameterization for polynomial curve approximation [44]. In this paper, we synthesize a dataset and train a fully convolutional network(FCN) [31] to solve the boundary correspondence problem for domain parameterization.

3 Method

In this work, the boundary of the given physical domain is represented by a planar polygon P with N vertices $\{p_0, \ldots, p_{N-1}\}$ in counter-clockwise order. Our task is to pick four vertices from the input boundary polygon that correspond to the four corner points of the parametric domain–the unit square. To utilize neural network technique, we need to extract the geometric features of the polygon P as the input of the network first. Under the assumption that the vertices $\{p_i\}_{i=0}^{N-1}$ are almost uniformly distributed on the boundary polygon, we can then define the exterior angle α_i $(-\pi < \alpha_i < \pi)$ at each vertex p_i $(i = 0, 1, \ldots, N-1)$ as the geometric features of the polygon P, where a positive angle indicates that the boundary bends inward and a negative one indicates that the boundary bends outward. An example is shown in Fig. 2. However, when the input vertices are not evenly distributed, we can add the length l_i between two subsequent vertices p_i and p_{i+1} as another feature.

The input of our neural network is an N-dimensional vector $x \in (-\pi, \pi)^N$, the i-th element of which is the exterior angle of p_i, $i = 0, 1, \ldots, N-1$. The output is also an N-dimensional vector y whose i-th element is the probability of p_i being a corner vertex, $i = 0, 1, \ldots, N-1$. Our goal is to find a map $f : (-\pi, \pi)^N \mapsto [0, 1]^N$ represented by the network such that the given data is well-fitted. After the network is trained (that is, f is computed), the four corner vertices can be obtained by finding the local maximums of $y = f(x)$. We elaborate the details in the rest of this section.

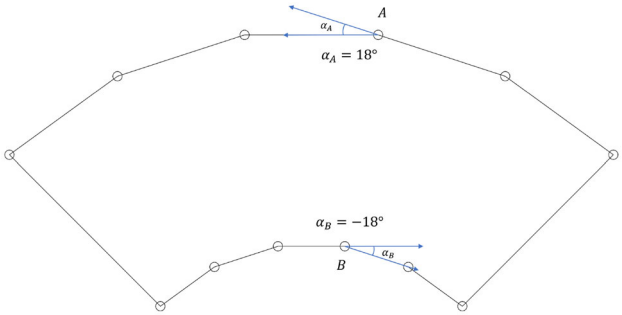


Fig. 2 An example of exterior angles in a polygon. The angle α_A at point A is a positive exterior angle and the angle α_B at point B is a negative one

3.1 Circular Invariance Property

Before describe the structure of our neural network, we first discuss the property of circular invariance, which is a desired property in our problem.

Definition 3.1 Let f be a map which sends an N-dimensional vector $x = (x_0, x_1, \ldots, x_{N-1})^T$ to an N-dimensional vector $y = (y_0, y_1, \ldots, y_{N-1})^T$. Let $x^{(k)} = (x_k, ..., x_{N-1}, x_0, ..., x_{k-1})^T$ and $y^{(k)} = (y_k, ..., y_{N-1}, y_0, ..., y_{k-1})^T$, $k = 0, 1, \ldots, N-1$. f is said to satisfy the property of circular invariance if $f(x^{(k)}) = y^{(k)}$, $k = 0, 1, \ldots, N-1$.

In our scenario, this property means the method should pick the same corner vertices, no matter which vertex of the boundary polygon is taken as the starting vertex. Obviously, if each layer of the network can maintain circular invariance, then the entire network will also have such invariance. Similar to the translation invariance property illustrated in [1], convolutional layers are considered more robust to circular invariance compared to fully connected layers, although perfect circular invariance is not possible due to down-sampling operations in the network. Here we make some analysis about the circular invariance property for different layers.

1. *Fully connected layers whose input layer and output layer have the same size* For an input vector $x \in \mathbb{R}^N$, the output vector is

$$y = Wx + b, \tag{3.1}$$

where $W \in \mathbb{R}^{N\times N}$ and $b \in \mathbb{R}^N$. By the circular invariance property, $y^{(k)} = Wx^{(k)} + b$, $k = 0, 1, \ldots, N-1$. Since $y^{(k)} = P^{k-1}y$ and $x^{(k)} = P^{k-1}x$, where

$$P = \begin{pmatrix} 0\,1\,0\cdots 0 \\ 0\,0\,1\cdots 0 \\ \vdots\,\vdots\,\vdots\,\ddots\,\vdots \\ 0\,0\,0\cdots 1 \\ 1\,0\,0\cdots 0 \end{pmatrix},$$

we have

$$P^{k-1}(Wx+b) = WP^{k-1}x + b.$$

Thus

$$P^{k-1}W = WP^{k-1}, \quad P^{k-1}b = b.$$

Therefore, W is a circular matrix

$$W = \begin{pmatrix} w_0 & w_1 & w_2 \cdots & w_{N-1} \\ w_{N-1} & w_0 & w_1 \cdots & w_{N-2} \\ w_{N-2} & w_{N-1} & w_0 \cdots & w_{N-3} \\ \vdots & \vdots & \vdots \ddots & \vdots \\ w_1 & w_2 & w_3 \cdots & w_0 \end{pmatrix} \tag{3.2}$$

and $b = c(1, 1, \ldots, 1)^T$ for some constant c.

2. *Convolutional layers* We focus on the convolution operation between two certain channels with circular padding. For an input vector $x = (x_0, x_1, \ldots, x_{N-1})^T$, the output is a vector $y = (y_0, y_1, \ldots, y_{N-1})$ with

$$y_i = \sum_{j=0}^{n-1} w_j x_{i-p+j} + c, \qquad i = 0, 1, \ldots, N-1, \tag{3.3}$$

where n is the kernel size, p is the padding size and c is the bias, and the subscripts of x are modulo N, for example, $x_{-1} := x_{N-1}$. Convolutional layer is a special case of fully connected layer, where the first row of the circular matrix W is $(w_p, \ldots, w_{n-1}, 0, \ldots, 0, w_0, \ldots, w_{p-1})$. Thus, the convolutional layers are circular invariant. Similarly, transposed convolutional layers are also circular invariant.

3. *Down-sampling and up-sampling layers* In our definition, circular invariance is defined only between input layers and output layers of same size. Therefore, our definition of circular invariance is not suitable for down-sampling and up-sampling operations. However, we can extend our definition for up-sampling but not down-sampling. Let s be the step size for up-sampling. If $f(x^{(k)}) = y^{(sk)}$, $k = 0, 1, \ldots, N-1$, then the up-sampling operation is called circular invariant. Let's look at an example. Suppose that the up-sampling is linear interpolation with step size $s = 2$. For an input vector $x = (x_0, x_1, \ldots, x_{N-1})$, the output is $y = (x_0, (x_0 + x_1)/2, x_1, \ldots, (x_{N-2} + x_{N-1})/2, x_{N-1}, (x_{N-1} + x_0)/2)$. It is easy to check that $y^{(2k)} = f(x^{(k)})$ for $k = 0, 1, \ldots, N-1$. That is, we still have circular invariance property. For down-sampling, circular invariance generally doesn't hold. Take the simplest case–average pooling with step size $s = 2$ and kernel size $n = 2$ as an example. As shown in Fig. 3, for a circular input $x = (1, -1, 2, -2, 3, -3)$, we can get two different outputs for the down-sampling

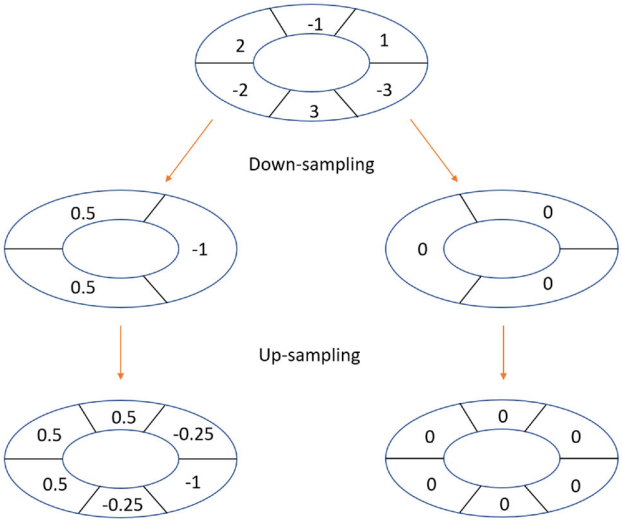


Fig. 3 An illustration of circular invariance property being not preserved in down-sampling operation

layer: $y^{(0)} = (0, 0, 0)$ and $y^{(1)} = (0.5, 0.5, -1)$ if we start the operation from x_1. After up-sampling operations for $y^{(0)}$ and $y^{(1)}$, we obtain $z^{(0)} = (0, 0, 0, 0, 0, 0)$ and $z^{(1)} = (0.5, 0.5, 0.5, -0.25, -1, -0.25)$, respectively. Obviously, circular invariance property gets lost due to down-sampling operation. However, if the input data x is continuous, that is, $|x_i - x_j|$ is small for small $|i - j|$ (remember here the indices are modulo N), then the output vector y after down-sampling is also continuous. One can easily show that $f(x^{(k)}) \approx y^{([k/s])}$. Therefore, circular invariance can be approximately preserved for small step size s.

3.2 Network Architecture

The proposed network is a 1D fully convolutional network(FCN) as shown in Fig. 4. The main purpose of using FCN rather than general CNN is to ensure (approximate) circular invariance property, since FCN contains only convolutional layers, transposed convolutional layers and activation layers. The input is a vector whose elements are the exterior angles of the vertices of the given boundary polygon. The output is a vector of the same size, each of whose elements is the possibility of the corresponding vertex being a corner vertex. The network is modified from UNet [42] and it contains a contracting path (on the left) and an expansive path (on the right). The contracting path consists of 5 convolutional layers with circular padding (shown in Fig. 5) and kernel sizes 102, 52, 52, 26 and 26 respectively. Down sampling is also performed in each layer with a step size 2. The expansive path consists of 5 transposed convolution layers with kernel sizes 26, 26, 52, 52 and 102, respectively, and up-sampling is performed in each layer using circular padding (shown in Fig. 5). Between the contacting path and the expansive path is a convolution with a kernel size 11. In order to prevent the degradation of the network, we refer to the idea of ResNet [15] and UNet [42] to add

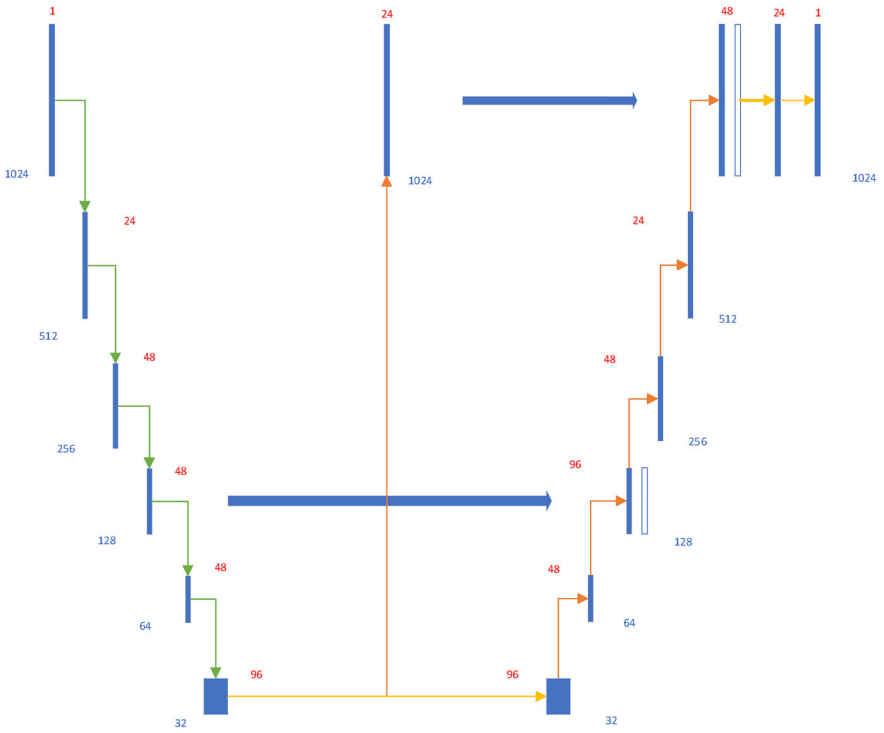


Fig. 4 The structure of our network. The green arrows indicate convolutional layers with down-sampling, the red arrows refer to transposed convolutional layers, the yellow arrows are convolutional layers with step size= 1, and the blue arrows mean concatenate. The numbers in red refer to the channel sizes of the data

two jump connections to the network. In the end, we use two convolution layers with a kernel size 1 to implement the role of the linear layer in general CNN. For each convolutional layer, a ReLU activation function is followed except the last one. In the network, we choose large kernels (the maximal size of which is 102) because they can extract features better, and large kernels do not lead to a surge in calculations in one-dimensional case. Finally, the output vector y is normalized such that $\max_i y_i = 1$.

3.3 Loss Function

The geometric nature of the problem requires that vertices near the corner vertices should have high possibility, while vertices far from the corner vertices should have low possibility. Therefore, the traditional one hot encoding is not appropriate for this task. We will design a special loss function for our purpose.

Given an input boundary polygon, we first compute a parametric representation $p(t)(0 \leq t \leq 1)$ for the polygon by using chord-length parameterization. Suppose that the four corner vertices of the polygon are $C := \{p(t_1), p(t_2), p(t_3), p(t_4)\}$, $0 \leq t_1 < t_2 < t_3 < t_4 \leq 1$. We construct a probability target function $g(t)(0 \leq t \leq 1)$

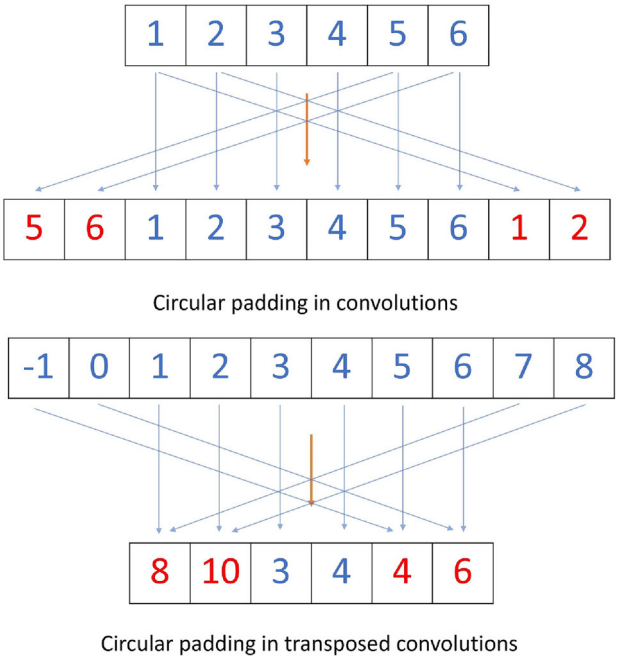


Fig. 5 An illustration of circular padding in (transposed) convolutions

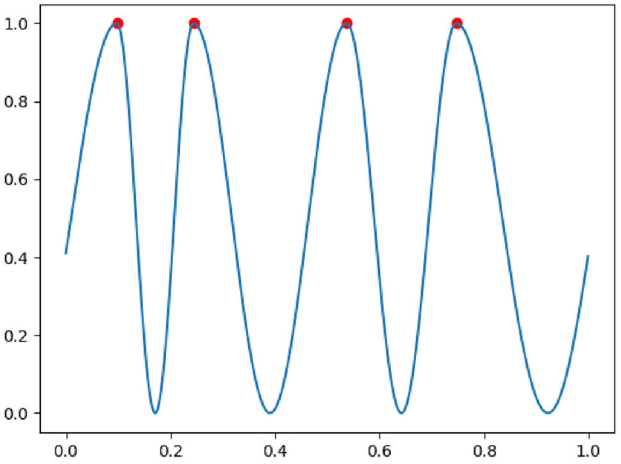


Fig. 6 Target probability function

which attains maximum values at $t = t_1, t_2, t_3, t_4$ by piecing four cosine functions smoothly together(see Fig. 6 for an illustration).

Now the difference between the output of our network $y \in \mathbb{R}^N$ and the probability function $g(t)$ is taken as the loss function:

$$L(y, g) = \frac{1}{N} \sum_{i=0}^{N-1} \left(y_i - g(\tilde{t}_i)\right)^2, \tag{3.4}$$

where $\tilde{t}_i$ is the parameter corresponding to the vertex p_i of the input polygon, $i = 0, 1, \ldots, N-1$.

3.4 Post Processing

The output of our network is a vector $y \in \mathbb{R}^N$ that indicates the probability of each vertex of the input polygon being a corner vertex. However, since the output data (a discrete function) is not smooth and fair enough and the local maximum values of the discrete function is not necessarily four, a post processing step is needed to extract four corner vertices as the final output of the method.

We first smooth the discrete function by some classic technique such as cubic spline fitting with curvature regularization. If the smoothed function has exactly four peaks at $t = t_1, t_2, t_3, t_4$, respectively, then the corresponding vertices $p(t_1)$, $p(t_2)$, $p(t_3)$, $p(t_4)$ are chosen as the corner vertices. However, more or less number of maximum values of the output function could appear. If the output function has only one maximum value at $t = t_1$, we just sample four uniform vertices (including the vertex $p(t_1)$ along the input polygon as the corner vertices. If the output function has two maximum values at $t = t_1$ and $t = t_2$, then it also has two minimum values between the two maximum values at $t = t_3$ and $t = t_4$, and we choose $p(t_i)$, $i = 1, 2, 3, 4$ as the corner vertices. However, from our experiments, we didn't find any output function with one or two maximum values, that is, the number of maximum values is at least three.

In the case that the output function has three maximum values at $t = \tilde{t}_i, \tilde{t}_j, \tilde{t}_k (i < j < k)$, the three corresponding vertices $p(\tilde{t}_i)$, $p(\tilde{t}_j)$, $p(\tilde{t}_k)$ will be certainly chosen as the corner vertices, and we need to find one more vertex to be the fourth corner vertex. To do so, we divide the input boundary into three parts(segments):

$$S_1 = \{p(\tilde{t}_{i+1}), p(\tilde{t}_{i+2}), \ldots, p(\tilde{t}_{j-1})\}, \qquad S_2 = \{p(\tilde{t}_{j+1}), p(\tilde{t}_{j+2}), \ldots, p(\tilde{t}_{k-1})\},$$
$$S_3 = \{p(\tilde{t}_{k+1}), \ldots, p(\tilde{t}_{N-1}), p(\tilde{t}_0), \ldots, p(\tilde{t}_{i-1})\}.$$

Let A_m be the sum of the exterior angles of all the vertices in S_m, and L_m be the length of the segment S_m, $m = 1, 2, 3$. L is the total length of the boundary polygon. Now suppose that m_0 is the index m such that $(A_m + \alpha L_m / L)$ is maximized (here α is a constant and we set $\alpha = 1$ in the following test). Then we choose the fourth corner vertex p_l on S_{m_0} such that p_l has a large exterior angle and it is distant from the other corner vertices. That is, we choose the fourth corner vertex p_l on a segment S_{m_0} that has large exterior angle change, and at the same time S_{m_0} is not too short. Some examples of this case are shown in Fig. 7.

In the case that the output function has more than four maximum values $\{y_m\}_{m=1}^M$ at $\{t_m\}_{m=1}^M$ respectively, without loss of generality, we assume that $y_m \geq 0.5$, $m =$

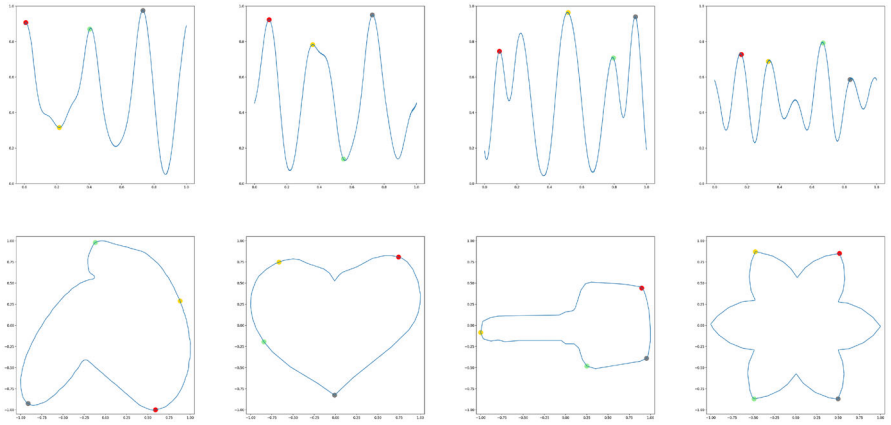


Fig. 7 More or less than four maximum values appear in the output function. The first row shows the output functions and the second row shows the corresponding boundary polygons, where points marked with different colors are the corner vertices we finally choose

$1, \ldots, \tilde{M}$ ($\tilde{M} \leq M$). If $\tilde{M} \leq 4$, let y_i, y_j, y_k, y_l be the four largest values among $\{y_m\}_{m=1}^{M}$, then the four corner vertices are chosen as $p(t_i)$, $p(t_j)$, $p(t_k)$, $p(t_l)$. If $\tilde{M} > 4$, we choose corner vertices from the vertex set $\{p(t_m)\}_{m=1}^{\tilde{M}}$ as follows. For each set of four vertices $\{p(t_i), p(t_j), p(t_k), p(t_l)\}$, $1 \leq i < j < k < l \leq \tilde{M}$, we divide the input boundary polygon into four segments S_1, S_2, S_3, S_4. Let A_m be the summation of the exterior angles of the vertices in S_m, $m = 1, 2, 3, 4$, and A_0 be the summation of the absolute values of the exterior angles at $\{p(t_i), p(t_j), p(t_k), p(t_l)\}$. Then we choose the set of four vertices $\{p(t_i), p(t_j), p(t_k), p(t_l)\}$ that minimize the sum $A_1 + A_2 + A_3 + A_4 - \beta S_0$ (here β is a constant and we set $\beta = 1$ in the following test) as the corner vertices. The idea behind is that we wish each boundary segment S_m has small exterior angle change, and the term $-\beta A_0^{i,j,k,l}$ is to avoid the cancellation of large positive and negative exterior angles on each segment S_m. Some examples of this case are also shown in Fig. 7.

The above process may miss some salient feature vertices (vertices with large exterior angles) as the corner vertices. To solve the problem, we can adjust a corner vertex to some feature vertex nearby.

3.5 Training Dataset Generation

We take the boundaries (simple closed polygons) of MPEG-7 dataset (977 data in total) as the inputs of the training data. The output for each input polygon is four corner vertices. We generate the output corner vertices for each input boundary through the following procedure. First, we apply the method for computing boundary correspondence between an input polygon and a unit square proposed in [62] to generate 20 sets of candidate corner vertices (each set contains four vertices). Then we randomly generate 480 sets of four tuple vertices from the input boundary. For each of these 500 sets of four tuple vertices, we regard it as the set of four corner vertices and

compute the corresponding parameterization of the input domain by employing the domain parameterization method put forward in [38]. We then calculate the sum of normalized angular and area distortion of the parameterization. The set of four tuple vertices with the smallest sum is taken as the final corner vertices for the given input boundary.

To further improve the results, we train our neural network using the above dataset. For each input boundary, the network produces a new set of corner vertices. If the new set of corner vertices differ from the previously labeled set of corner vertices, we again apply the domain parameterization method proposed in [38] to compare their distortion. If the new set of corner vertices have smaller distortion, the set of labeled corner vertices are updated with the new set of corner vertices. This process is repeated until the labeled set of corner vertices become stable. Experiments find that two iterations are enough to get a stable labeled set of corner vertices.

3.6 Training Process

In the training process, we use Adam optimizer with a batch size 16, a learning rate 10^{-4} and weight decay 0.0003. We train the network for 80 epochs. The training process takes 17.5 minutes on an Nvidia GTX-1650 GPU, Intel i7-9750H CPU computer with 16 GB memory.

4 Evaluation

In this section, we evaluate the performance of our neural network through two datasets: MPEG-7 and CAD dataset and validate the circular invariance property by experimental examples.

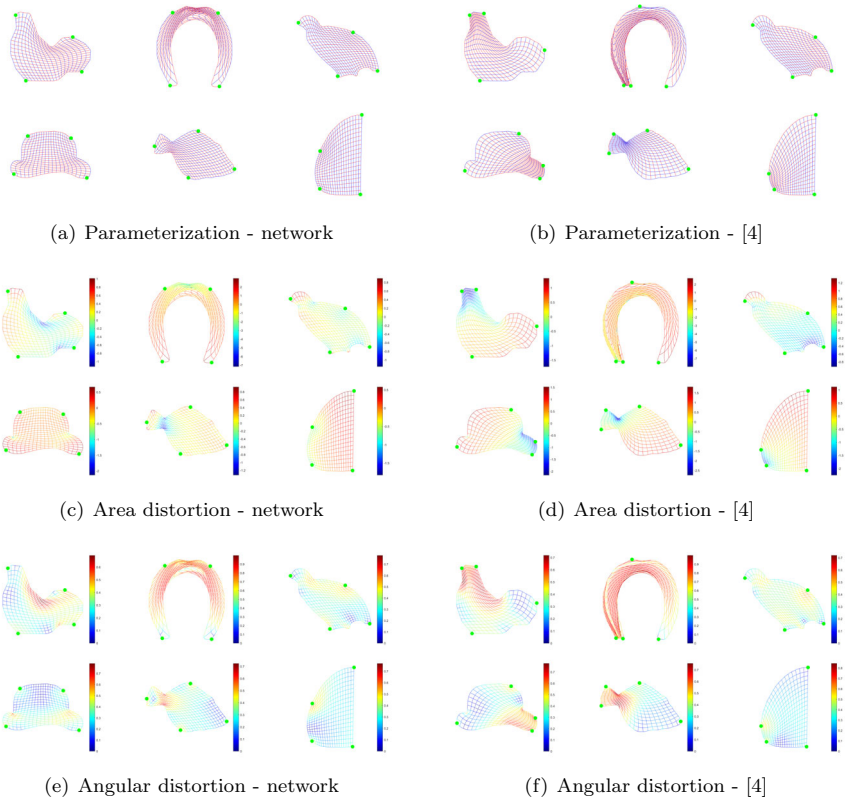
4.1 Cross Validation on the MPEG-7 Dataset

Given an input polygonal domain, our neural network produces four corner vertices from which boundary correspondence between the boundary of the input domain and the boundary of the unit square can be established. Based upon the boundary correspondence, the parameterization of the input domain, i.e., the map from the unit square to the input domain can be computed [38]. The performance of our neural network can be measured by the quality of the parameterization, i.e., the angular and area distortion and the success(bijectivity) rate of the parameterization. The angular distortion of the mapping is measured by the modulus of its Beltrami coefficient μ [38] with the best value $\mu = 0$, and the area distortion is characterized by the determinant of the Jacobian J of the map with the best value $|J| = 1$. J_{std} reflects the uniformity of area distortion.

We first cross-validate our model on the dataset MPEG-7 we construct in Sect. 3.5. Specifically, we divide the data set into 10 units for training set, testing set and validation set according to the ratio $7 : 2 : 1$, and train 10 networks using different parts of the data, with each unit being the validation set exactly once. A comparison is performed

Table 1 Comparison between our network and the optimization method [62] on MPEG-7 dataset

	$log(\frac{J_{\max}}{J_{\min}})$	$\mu_{\max}$	J_{std}	μ_{ave}	Bijectivity	Time(s)
Our model	3.644	0.825	0.719	0.448	93.8%	0.010
Zheng's method	4.188	0.786	0.897	0.476	90.2%	10.234



(a) Parameterization - network (b) Parameterization - [4]

(c) Area distortion - network (d) Area distortion - [4]

(e) Angular distortion - network (f) Angular distortion - [4]

Fig. 8 Some results of domain parameterization on MPEG-7 data set using the corner vertices generated by our network and the method in [62]

between the average behavior of the ten models over each testing set and the results by the optimization based method [62] on the whole dataset. As shown in Table 1, our neural network model is not only much faster than the traditional method [62], but also achieves a higher bijectivity rate and better parameterization quality in terms of average angular distortion and uniformity of area distortion. The traditional method achieves a smaller maximum angular distortion, which can be explained by the nature of the method, that is, it mainly focuses on the local geometric details. On the contrary, our model pays more attention on the global behavior of the parameterization. Some parameterization results by our method and the method in [62] are shown in Fig. 8.

Table 2 Comparison between our network and the optimization method [62] on CAD dataset

	$log(\frac{J_{\max}}{J_{\min}})$	$\mu_{\max}$	J_{std}	μ_{ave}	Bijectivity	Time(s)
Our network	4.774	0.938	0.842	0.232	95.5%	0.010
Zheng's method	4.744	0.938	1.035	0.244	94.6 %	9.688

4.2 Results on CAD Dataset

In addition to the MPEG-7 dataset, we collect some three-dimensional CAD models and project them onto two-dimensional planes to obtain a set of 2D domains with 1228 data (we refer it to a CAD dataset). We test our network on the CAD dataset to check for the generalization capacity of the network, and the statistical results are reported in Table 2. From the statistics, we can see that our neural network exhibits quite good generalization capacity. We also compare the results of our network with those by the optimization method [62]. The two methods produce about the same parameterization quality, but our method achieves smaller average angular distortion and more uniform area distortion. Furthermore, our method gives higher bijectivity rate of the parameterization and is much faster. Some parameterization examples by the two methods are shown in Fig. 9.

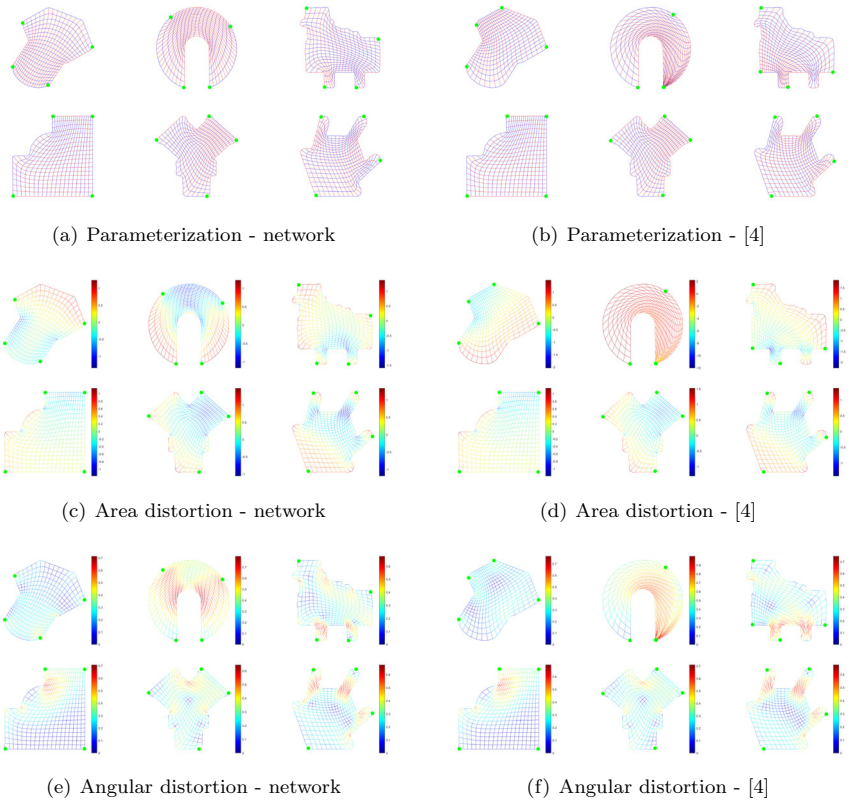
4.3 Validation of Circular Invariance Property

We further validate the circular invariance property of our network through experimental examples. In our problem, circular invariance means that our method should pick the same corners, no matter which vertex of the boundary polygon is taken as the beginning vertex.

For a given polygon $P = \{p_0, ..., p_{N-1}\}$, let $P^{(k)} = \{p_k, ..., p_N, p_0, ..., p_{k-1}\}$, $k = 0, 1, \ldots, N-1$, that is, $P^{(k)}$ is the same polygon but with different starting vertex. For each polygon P, we use our neural network to compute N sets of corner vertices $C^{(k)} := \{p_1^k, p_2^k, p_3^k, p_4^k\}$ with each $P^{(k)}$ being the input polygon, $k = 0, 1, \ldots, N-1$. Then the circular invariance property holds if all the sets $C^{(k)}$ are the same. However, since our neural network can't guarantee exact circular invariance due to down-sampling, the point sets $C^{(k)}$ can't be the same in practice. If the difference between the point sets is small, the circular invariance property holds approximately. To quantify the difference between different corner sets, we introduce the following definition.

Definition 4.1 The distance $d(p_i, p_j)$ between two vertices p_i, p_j of a given polygon $P = \{p_0, ..., p_{N-1}\}$ is the length of the shortest path from p_i to p_j or p_j to p_i along the polygon. The distance between two sets of corner vertices $C^{(k)}$ and $C^{(l)}$ is defined as

$$d(C^{(k)}, C^{(l)}) := \min_{0 \le j \le 3} \frac{1}{4} \sum_{i=1}^{4} d(p_i^k, p_{i+j}^l), \tag{4.1}$$



(a) Parameterization - network (b) Parameterization - [4]

(c) Area distortion - network (d) Area distortion - [4]

(e) Angular distortion - network (f) Angular distortion - [4]

Fig. 9 Some results of domain parameterization on CAD data set based on the boundary correspondences by our network and the method in [62]

where the subscript j of p_j is modulo 5. That is, the distance between two sets of points is the average distance between the pairwise points in the two sets. The relative difference between N sets of corner vertices $C^{(k)}$, $k = 0, 1, \ldots, N-1$ is defined by

$$Diff\left(\{C^{(k)}\}_{k=0}^{N-1}\right) := \frac{2}{N(N-1)} \sum_{0 \le i < j \le N-1} \frac{1}{L} d(C^{(i)}, C^{(j)}), \tag{4.2}$$

where L is the total length of the input boundary polygon.

Now for each polygon in the MPEG-7 dataset, we compute the relative difference between N sets of corner vertices according to (4.2). The statistics including the maximum, minimum, average and the standard deviation of the differences are reported in Table 3. Figure 10 shows the histogram of the distribution of the differences, while Fig. 11 presents a visualization of the distribution of corner vertices. Notice that the middle bottom picture has five corner vertices, and it means that any four out of the five vertices can be served as the corner vertices for each instance. This is the possible unstable case and it explains why relatively large differences between the sets of corner

Table 3 Statistics on the differences of the sets of corner vertices with different starting vertices of input boundary polygons

Difference	Max	Min	Average	Standard deviation
Value	0.0758	0.0000	0.0111	0.0178

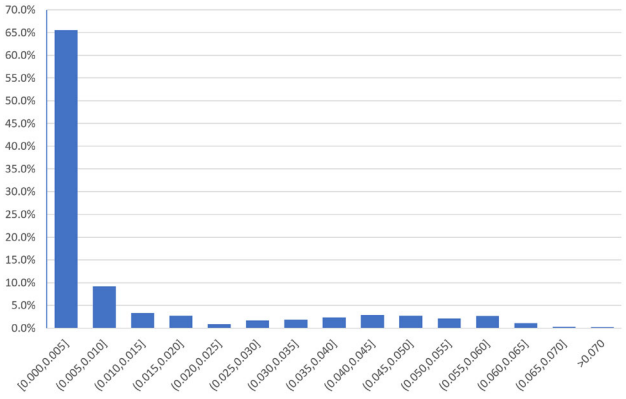


Fig. 10 Histogram of the distribution of the differences of the set of corner vertices with different start vertices

vertices could occur as illustrated in Fig. 10. However, this situation is reasonable in practice. In summary, our neural network preserves the circular invariance property approximately in practical examples, and it has little influence on the results of corner vertices selection no matter which vertex is chosen as the starting vertex of a given input boundary polygon.

5 Conclusion

Based on a fully convolutional neural network, we present a new automatic algorithm for calculating the boundary correspondence in the planar domain parameterization problem. Compared with the traditional optimization based approach, our method is faster by several orders of magnitude and results in better parameterization quality. Furthermore, our neural network exibits good generalization capacity on different datasets and it preserves circular invariance property approximately–a very important property for our problem.

The limitation of our method is that the capacity of the network depends much on the training data. Currently, we apply the traditional optimization based algorithm to generate most of the ground-truth data for training, which may limit the effectiveness of the network. On the other hand, the size of the data set is relatively small, which may influence the generalization capacity of our neural network. We guess that enlarging the data set would enhance the capacity and effectiveness of our method. Furthermore, our network is only suitable for setting up boundary corre-

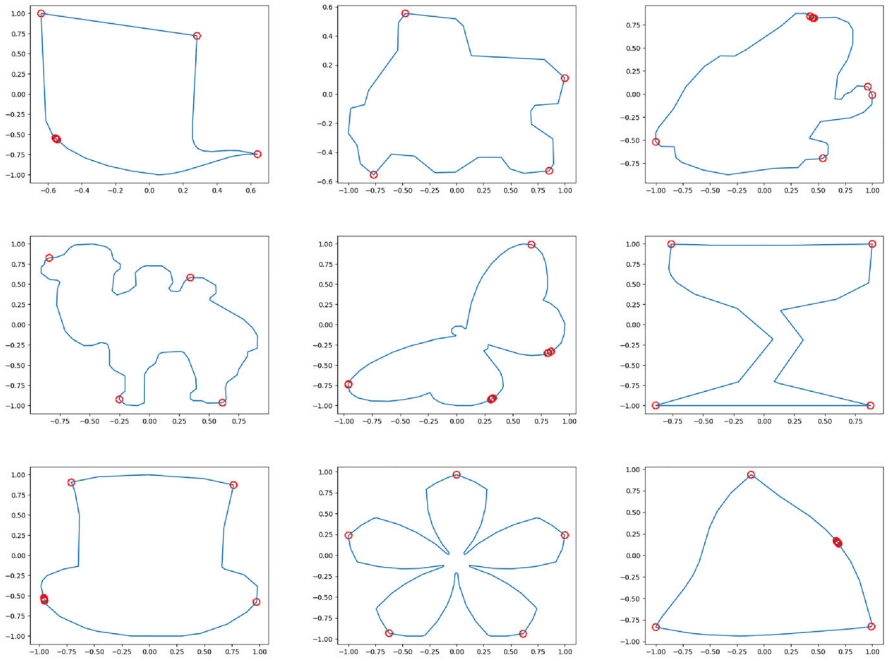


Fig. 11 Visualization of the sets of corner vertices for different starting vertices

spondence for two-dimensional domain parameterization. Generalizing our method to three-dimensional domains would be a direction of our future research. A simple idea is to use graph convolutional networks to tackle the problem.

Acknowledgements This work is supported by NSF of China (no.61972368).

References

1. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations? J. Mach. Learn. Res. **20**(184), 1–25 (2019)
2. Blidia, A., Mourrain, B., Xu, G.: Geometrically smooth spline bases for data fitting and simulation. Computer Aided Geometric Des. **78**, 87 (2020)
3. 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, pp. 7213–7222 (2019)
4. Buchegger, F., Jüttler, B.: Planar multi-patch domain parameterization via patch adjacency graphs. Computer-Aided Des. **82**, 2–12 (2017)
5. Chen, L., Xu, G., Wang, S., Shi, Z., Huang, J.: Constructing volumetric parameterization based on directed graph simplification of $\ell 1$ polycube structure from complex shapes. Computer Methods Appl. Mech. Eng. **351**, 422–440 (2019)
6. Cohen, E., Martin, T., Kirby, R.M., Lyche, T., Riesenfeld, R.F.: Analysis-aware modeling: understanding quality considerations in modeling for isogeometric analysis. Computer Methods Appl. Mech. Eng. **199**(5), 334–356 (2010)

7. 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, pp. 5868–5877 (2017)
8. 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, pp. 3844–3852. Curran Associates Inc. (2016)
9. Farin, G., Hansford, D.: Discrete coons patches. Computer Aided Geometric Des. **16**(7), 691–700 (1999)
10. Fey, M., Lenssen, J.E., 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, pp. 869–877 (2018).
11. Gao, Z., Yan, J., Zhai, G., Zhang, J., Yang, Y., Yang, X.: Learning Local Neighboring Structure for Robust 3D Shape Representation (2020)
12. Gong, S., Chen, L., Bronstein, M., Zafeiriou, S.: Spiralnet++: A fast and highly efficient mesh convolution operator. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 4141–4148 (2019).
13. Gravesen, J., Evgrafov, A., Nguyen, M., NØrtoft, P.: Planar parametrization in isogeometric analysis. In: Mathematical Methods for Curves and Surfaces, pp. 189–212. Springer (2014)
14. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: MeshCNN: a network with an edge. ACM Trans. Graph. **38**(4), 90–19012 (2019)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. IEEE, Las Vegas, NV, USA (2016).
16. Hinz, J., Möller, M., Vuik, C.: Elliptic grid generation techniques in the framework of isogeometric analysis applications. Computer Aided Geometric Des. **65**, 48–75 (2018)
17. Hu, K., Zhang, Y.J.: Centroidal Voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. Computer Methods Appl. Mech. Eng. **305**, 405–421 (2016)
18. Hu, K., Zhang, Y.J., Liao, T.: Surface segmentation for polycube construction based on generalized centroidal Voronoi tessellation. Computer Methods Appl. Mech. Eng. **316**, 280–296 (2017)
19. Hughes, T.J.R., Cottrell, J.A., Bazilevs, Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. Computer Methods Appl. Mech. Eng. **194**(39), 4135–4195 (2005)
20. Ji, Y., Yu, Y.-Y., Wang, M.-Y., Zhu, C.-G.: Constructing high-quality planar NURBS parameterization for isogeometric analysis by adjustment control points and weights. J. Comput. Appl. Math. **396**, 113615 (2021)
21. Kapl, M., Sangalli, G., Takacs, T.: Dimension and basis construction for analysis-suitable G(1) two-patch parameterizations. Computer Aided Geometric Des. **52–53**, 75–89 (2017)
22. Kapl, M., Buchegger, F., Bercovier, M., Juttler, B.: Isogeometric analysis with geometrically continuous functions on planar multi-patch geometries. Computer Methods Appl. Mech. Eng. **316**, 209–234 (2017)
23. Kapl, M., Sangalli, G., Takacs, T.: Construction of analysis-suitable G(1) planar multi-patch parameterizations. Computer Aided Des. **97**, 41–55 (2018)
24. Kapl, M., Sangalli, G., Takacs, T.: An isogeometric C-1 subspace on unstructured multi-patch planar domains. Computer Aided Geometric Des. **69**, 55–75 (2019)
25. Litany, O., Remez, T., Rodolá, E., Bronstein, A., Bronstein, M.: Deep functional maps: Structured prediction for dense shape correspondence. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5660–5668 (2017).
26. Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., van der Laak, J.A.W.M., van Ginneken, B., Sánchez, C.I.: A survey on deep learning in medical image analysis. Med. Image Anal. **42**, 60–88 (2017)
27. Liu, L., Zhang, Y., Hughes, T., Scott, M., Sederberg, T.: Volumetric T-spline construction using Boolean operations. Eng. Computers **30**(4), 425–439 (2014)
28. Liu, L., Zhang, Y., Liu, Y., Wang, W.: Feature-preserving T-mesh construction using skeleton-based polycubes. Computer-Aided Des. **58**, 162–172 (2015)
29. Liu, H., Yang, Y., Liu, Y., Fu, X.-M.: Simultaneous interior and boundary optimization of volumetric domain parameterizations for iga. Computer Aided Geometric Des. **79**, 101853 (2020)
30. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: a survey. Int. J. Computer Vision **128**, 87 (2020)

31. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440 (2015).
32. Mantzaflaris, A., Jüttler, B., Khoromskij, B.N., Langer, U.: Low rank tensor methods in Galerkin-based isogeometric analysis. Computer Methods Appl. Mech. Eng. **316**, 1062–1085 (2017)
33. Minaee, S., Boykov, Y.Y., Porikli, F., Plaza, A.J., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 1 (2021)
34. Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J.: Deep learning-based text classification: a comprehensive review. ACM Comput. Surv. **54**, 3 (2021)
35. Monti, F., Boscaini, D., Masci, J., Rodolá, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5425–5434 (2017).
36. Nguyen, T., Jüttler, B.: Parameterization of contractible domains using sequences of harmonic maps. In: Curves and Surfaces. Lecture Notes in Computer Science, pp. 501–514. Springer, Berlin, Heidelberg (2010).
37. Nian, X., Chen, F.: Planar domain parameterization for isogeometric analysis based on teichmüller mapping. Computer Methods Appl. Mech. Eng. **311**, 41–55 (2016)
38. Pan, M., Chen, F., Tong, W.: Low-rank parameterization of planar domains for isogeometric analysis. Computer Aided Geometric Des. **63**, 1–16 (2018)
39. Pauley, M., Nguyen, D.-M., Mayer, D., Špeh, J., Weeger, O., Jüttler, B.: The Isogeometric Segmentation Pipeline. In: Jüttler, B., Simeon, B. (eds.) Isogeometric Analysis and Applications 2014. Lecture Notes in Computational Science and Engineering, pp. 51–72. Springer International Publishing, Cham (2015).
40. Pilgerstorfer, E., Juttler, B.: Bounding the influence of domain parameterization and knot spacing on numerical stability in Isogeometric Analysis. Computer Methods Appl. Mech. Eng. **268**, 589–613 (2014)
41. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in neural information processing systems. Curran Associates Inc, UK (2017)
42. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation (2015)
43. Sahillioğlu, Y.: Recent advances in shape correspondence. Visual Computer **36**, 871 (2020)
44. Scholz, F., Jüttler, B.: Parameterization for polynomial curve approximation via residual deep neural networks. Computer Aided Geometric Des. **85**, 101977 (2021)
45. Speleers, H., Manni, C.: Optimizing domain parameterization in isogeometric analysis based on Powell-Sabin splines. J. Comput. Appl. Math. **289**, 68–86 (2015)
46. van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. Computer Gr. Forum **30**(6), 1681–1707 (2011)
47. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph Attention Networks (2017)
48. Wang, W., Zhang, Y., Liu, L., Hughes, T.J.R.: Trivariate solid t-spline construction from boundary triangulations with arbitrary genus topology. Computer-Aided Design **45**(2), 351–360 (2013). Solid and Physical Modeling 2012
49. Wang, S.: Iga-suitable planar parameterization with patch structure simplification of closed-form polysquare. Computer Methods Appl. Mech. Eng. **392**, 114678 (2022)
50. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. Gr. **38**(5), 146–114612 (2019)
51. Wang, D., Xu, J., Gao, F., Wang, C.C.L., Gu, R., Lin, F., Rabczuk, T., Xu, G.: Iga-reuse-net: a deep-learning-based isogeometric analysis-reuse approach with topology-consistent parameterizationimage 1. Computer Aided Geometric Des. **95**, 102087 (2022)
52. 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 (ICCV), pp. 1042–1051 (2019).
53. Xiao, S., Kang, H., Fu, X.-M., Chen, F.: Computing IGA-suitable planar parameterizations by PolySquare-enhanced domain partition. Computer Aided Geometric Des. **62**, 29–43 (2018)
54. Xu, G., Mourrain, B., Duvigneau, R., Galligo, A.: Parameterization of computational domain in isogeometric analysis: methods and comparison. Computer Methods Appl. Mech. Eng. **200**(23), 2021–2031 (2011)

55. 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. **252**, 275–289 (2013)
56. Xu, G., Mourrain, B., Galligo, A., Rabczuk, T.: High-quality construction of analysis-suitable trivariate nurbs solids by reparameterization methods. Comput. Mech. (2014).
57. Xu, J., Chen, F., Deng, J.: Two-dimensional domain decomposition based on skeleton computation for parameterization and isogeometric analysis. Computer Methods Appl. Mech. Eng. **284**, 541–555 (2015)
58. Xu, G., Li, M., Mourrain, B., Rabczuk, T., Xu, J., Bordas, S.P.A.: Constructing IGA-suitable planar parameterization from complex CAD boundary by domain partition and global/local optimization. Computer Methods Appl. Mech. Eng. **328**, 175–200 (2018)
59. Xu, G., Li, B., Shu, L., Chen, L., Xu, J., Khajah, T.: Efficient r-adaptive isogeometric analysis with winslow's mapping and monitor function approach. J. Comput. Appl. Math. **351**, 186–197 (2019)
60. Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C.L., Hughes, T.J.R.: Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Computer Methods Appl. Mech. Eng. **196**(29), 2943–2959 (2007)
61. Zheng, Y., Chen, F.: Volumetric Boundary Correspondence for Isogeometric Analysis Based on Unbalanced Optimal Transport. Computer-Aided Des. **140**, 103078 (2021)
62. Zheng, Y., Pan, M., Chen, F.: Boundary correspondence of planar domains for isogeometric analysis based on optimal mass transport. Computer-Aided Des. **114**, 28–36 (2019)