Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

ARAP++: an extension of the local/global approach to mesh parameterization*

Zhao WANG[1], Zhong-xuan LUO‡[1,2], Jie-lin ZHANG[1], Emil SAUCAN[3]

([1]*School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China*)

([2]*School of Software, Dalian University of Technology, Dalian 116620, China*)

([3]*Max Planck Institute for Mathematics in the Sciences, Leipzig 70569, Germany*)

E-mail: wangzhao.ok@163.com; zxluo@dlut.edu.cn; jielinzh@dlut.edu.cn; semil@ee.technion.ac.il

Received June 11, 2015; Revision accepted Feb. 16, 2016; Crosschecked May 6, 2016

Abstract: Mesh parameterization is one of the fundamental operations in computer graphics (CG) and computer-aided design (CAD). In this paper, we propose a novel local/global parameterization approach, ARAP++, for single- and multi-boundary triangular meshes. It is an extension of the as-rigid-as-possible (ARAP) approach, which stitches together 1-ring patches instead of individual triangles. To optimize the spring energy, we introduce a linear iterative scheme which employs convex combination weights and a fitting Jacobian matrix corresponding to a prescribed family of transformations. Our algorithm is simple, efficient, and robust. The geometric properties (angle and area) of the original model can also be preserved by appropriately prescribing the singular values of the fitting matrix. To reduce the area and stretch distortions for high-curvature models, a stretch operator is introduced. Numerical results demonstrate that ARAP++ outperforms several state-of-the-art methods in terms of controlling the distortions of angle, area, and stretch. Furthermore, it achieves a better visualization performance for several applications, such as texture mapping and surface remeshing.

Key words: Mesh parameterization, Convex combination weights, Stretch operator, Jacobian matrix
http://dx.doi.org/10.1631/FITEE.1500184 **CLC number:** TP391

1 Introduction

Mesh parameterization is an important research topic in computer graphics (CG), and it has been widely used in digital geometry processing tasks, such as texture mapping (Haker *et al.*, 2000), surface fitting (Hormann and Greiner, 2000b), and surface remeshing (Hormann *et al.*, 2001). When a discrete surface is directly flattened onto the plane, distortions are inevitable due to the parameterization process. Preserving the geometric properties of the original mesh is essential for a good parameterization.

In this paper, we propose a novel local/global mesh parameterization approach, ARAP++. Our work is inspired mainly by the as-rigid-as-possible (ARAP) approach (Sorkine and Alexa, 2007; Liu *et al.*, 2008; Bouaziz *et al.*, 2012) and the convex combination approach (Eck *et al.*, 1995; Floater, 1997). ARAP++ adopts the idea of ARAP regarding the approximation of the Jacobian matrix with a fitting matrix, and then achieves the global flattened result by stitching together the local 1-ring patches. To optimize the local spring energy, we introduce the convex combination weights and stretch operator (Sander *et al.*, 2001; Yoshizawa *et al.*, 2004) to ARAP++. In consequence, ARAP++ renders lower area and stretch distortions than ARAP. The flattened results of our method are obtained using

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61432003, 61572105, 11171052, and 61328206)

ORCID: Zhao WANG, http://orcid.org/0000-0001-5659-9364

a free boundary. Therefore, ARAP++ outperforms those of the convex combination approach in processing boundaries of models. These facts show that ARAP++ enhances robustness and adaptiveness relative to both methods above, and achieves a better result in texture mapping (Fig. 1).

The main contributions of this paper are listed as follows:

1. We devise a novel local/global approach to mesh parameterization (ARAP++) based on the optimization of spring energy, and analyze the relation to the ARAP approach which is based on the optimization of the Dirichlet energy. In addition, a broader class of convex combination weights are considered in our method (Sections 4.2 and 4.3).

2. Compared with ARAP, ARAP++ improves the local phase and obtains the flattened results by stitching together the 1-ring patches, instead of individual triangles (Section 4.2). Moreover, we give a simple and fast calculation method to obtain an authalic fitting matrix (in the Appendix).

3. To deal with high-curvature models, we introduce a stretch operator to ARAP++. It enhances the robustness of the method, and eliminates the influence of overlapping and flipping (Section 4.6).

2 Related work

In the past decade, extensive research was conducted regarding the mesh parameterization problem. We refer readers to several survey papers for fundamental theories and methods (Floater and Hormann, 2005; Hormann *et al.*, 2007; Sheffer *et al.*, 2007). Below we briefly review the major techniques which have close relationship to our work.

The convex combination approach (Tutte, 1963; Eck *et al.*, 1995; Floater, 1997; 2003; Desbrun *et al.*, 2002; Lee *et al.*, 2002; Yoshizawa *et al.*, 2004) is a type of linear and fixed boundary parameterization. It is fast and stable without producing overlapping. However, if the boundary is not fixed optimally in advance, then it will lead to higher distortions along the boundary as well as the regions which are away from it.

Some methods rely on angle optimization, both for their intuitiveness and for their effectiveness. The angle-based flattening methods (Sheffer and de Sturler, 2001; Sheffer *et al.*, 2005; Kharevych *et al.*, 2006; Zayer *et al.*, 2007) are defined in the angle space, producing minimal angular distortion. Jin *et al.* (2008) built a unified framework for discrete surface Ricci flow algorithms. Chen *et al.* (2008) modified the Gaussian curvature by means of the transition probability matrix, and computed metric scaling as a solution of the Poisson equation. Weber and Zorin (2014) proposed an algorithm with arbitrarily fixed boundary, which guarantees that the result is locally injective.

The energy optimization methods (Hormann and Greiner, 2000a; Lévy *et al.*, 2002; Degener *et al.*, 2003; Weber *et al.*, 2012) based on the singular values of the Jacobian matrix also constitute an important part of commonly employed methods. Sorkine and Alexa (2007) and Jacobson *et al.* (2012) devised nonlinear energy for surface deformation, and minimized it by a combination of linear solvers and other operation (e.g., singular value decomposition (SVD)). Liu *et al.* (2008) presented a local/global algorithm

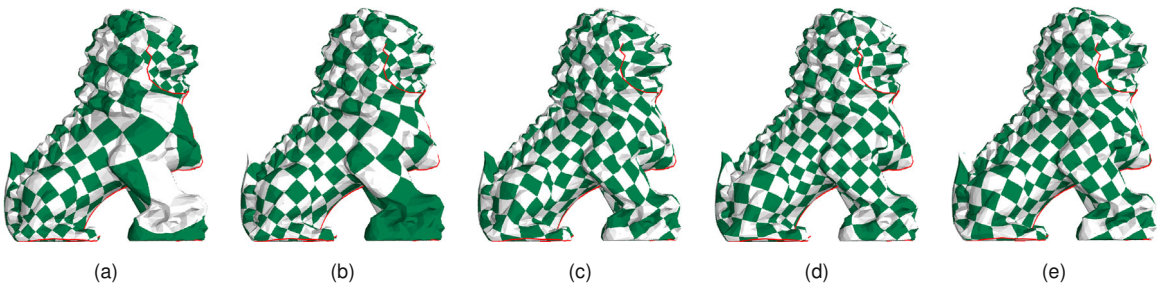

Fig. 1 Texture of different mesh parameterizations for a Chinese Dragon model (red lines represent the seams of the closed mesh when cut to a disk): (a) least squares conformal maps (LSCM) (Lévy *et al.*, 2002); (b) linear angle-based flattening (LABF) (Zayer *et al.*, 2007); (c) as-rigid-as-possible (ARAP) (Liu *et al.*, 2008); (d) bounded distortion as-rigid-as-possible (BD-ARAP) (Lipman, 2012); (e) ARAP++. References to color refer to the online version of this figure

which introduces the idea of ARAP into surface parameterization. Zhang *et al.* (2010) promoted the ARAP algorithm to meshless parameterization and mesh reconstruction. Lipman (2012) and Aigerman and Lipman (2013) dealt with the question of injectivity in bounded distortion mapping spaces, and improved many popular algorithms, such as BD-LSCM and BD-ARAP. Bouaziz *et al.* (2012) proposed a unified framework for geometry processing based on shape proximity function and shape projection operators. Levi and Zorin (2014) introduced the idea of strict minimizers to the ARAP approach as well as for other applications.

Moreover, there are a variety of parameterization methods. Hormann *et al.* (1999) provided a hierarchical representation of discrete surface, which includes two major steps: edge collapse and vertex split. Zigelman *et al.* (2002) presented multidimensional scaling (MDS) parameterization, which can preserve the geodesic distance of a triangular mesh. Gu and Yau (2002; 2003) approached the global surface parametrization problem by computing conformal structures of general two-manifolds. Gortler *et al.* (2006) described some simple properties of discrete one-forms, and their applications to three-dimensional (3D) mesh embedding. Chen *et al.* (2007) flattened surfaces with the theory of local tangent space alignment (LTSA). Mullen *et al.* (2008) defined discrete spectral conformal maps, and obtained a conformal mapping result by minimization of Dirichlet energy. Zhao *et al.* (2013) presented an authalic flattening method based on the optimal mass transport technique.

3 Preliminaries

In this section, we give a brief overview for the convex combination approach. The basic idea is to map the boundary nodes of a 3D mesh to a convex polygon in the plane. The internal node can be expressed as a weighted average of their 1-ring nodes, thus guaranteeing a bijective mapping.

Denote S as a 3D mesh, $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ the internal nodes, and $\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \dots, \boldsymbol{x}_N$ the boundary nodes of S. Denote S^* as a 2D mesh, $\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n$ the internal nodes, and $\boldsymbol{u}_{n+1}, \boldsymbol{u}_{n+2}, \dots, \boldsymbol{u}_N$ the boundary nodes of S^*. The internal node $\boldsymbol{u}_i$ can be represented as a strict convex combination of its 1-ring nodes $\boldsymbol{u}_j$ in the plane, satisfying

$$\boldsymbol{u}_i - \sum_{j=1}^{n} \omega_{i,j} \boldsymbol{u}_j = \sum_{j=n+1}^{N} \omega_{i,j} \boldsymbol{u}_j, \tag{1}$$

where $\omega_{i,j}$ represents the convex combination weight. The solutions of Eq. (1) are the coordinates of S^*.

3.1 Spring energy and Dirichlet energy

The convex combination approach can also be explained as follows. In a plane, a linear combination of vectors $\boldsymbol{u}_i = \sum_{j=1}^{d_i} \omega_{i,j} \boldsymbol{u}_j$ minimizes the spring energy (Hoppe *et al.*, 1993), which is a weighted sum of squared distances of all neighboring points $\boldsymbol{u}_j$ to $\boldsymbol{u}_i$, that is,

$$E(i)_{\text{spring}} = \sum_{j=1}^{d_i} \omega_{i,j} \|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2, \tag{2}$$

where d_i is the degree of $\boldsymbol{u}_i$. The optimization target in a plane is to minimize the global spring energy $\sum_{i=1}^{n} E(i)_{\text{spring}}$. Then we take the partial derivative of $\boldsymbol{u}_i$, and obtain Eq. (1) in another way.

Inspired by Desbrun *et al.* (2002), the Dirichlet energy (Pinkall and Polthier, 1993) over the whole 1-ring neighborhood is

$$E(i)_{\text{Dirichlet}} = \sum_{j=1}^{d_i} \cot \alpha_{i,j} \cdot \|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2, \tag{3}$$

where $\alpha_{i,j}$ is the angle opposite to the oriented edge $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the 3D 1-ring neighborhood. Just like various spring coefficients corresponding to the different spring energy, we can regard the Dirichlet energy as a special spring energy.

In addition, there are five classic types of weights $\omega_{i,j}$ that can be applied to Eq. (2), namely uniform (Tutte, 1963), shape-preserving (Floater, 1997), mean-value (Floater, 2003), cotan (Eck *et al.*, 1995), and intrinsic (Desbrun *et al.*, 2002) weights.

3.2 Error metrics on discrete surface

Let F denote the number of faces of S, θ and ϕ the angles of S and S^* respectively, and area(T) the surface area of T. Then the angle distortion and

area distortion can be written as follows:

$$\mathrm{Dist}_{\mathrm{angle}} = \frac{1}{3F}\sum_{j=1}^{F}\sum_{i=1}^{3}\left|\theta_{j,i}-\phi_{j,i}\right|,$$

$$\mathrm{Dist}_{\mathrm{area}} = \sum_{j=1}^{F}\left|\frac{\mathrm{area}(T_j)}{\sum_{i=1}^{F}\mathrm{area}(T_i)} - \frac{\mathrm{area}(T_j^*)}{\sum_{i=1}^{F}\mathrm{area}(T_i^*)}\right|.$$

We can measure the stretch distortion of the flattening results according to the stretch metric (Sander *et al.*, 2001). The Jacobian matrix $\boldsymbol{J}_f$ is obtained from the affine mapping f. The largest and smallest singular values of $\boldsymbol{J}_f$ are denoted by σ_1 and σ_2, respectively. Then the $L^2(T)$ stretch norm is defined over a triangle T as

$$L^2(T) = \sqrt{(\sigma_1^2+\sigma_2^2)/2}.$$

The norm $L^2(T)$ represents the root-mean-square stretch. Similarly, there is an analogous norm over the entire mesh as

$$L^2(S) = \sqrt{\frac{\sum_{i=1}^{F}\mathrm{area}(T_i^*)}{\sum_{i=1}^{F}\mathrm{area}(T_i)}}\sqrt{\frac{\sum_{i=1}^{F}(L^2(T_i))^2\mathrm{area}(T_i)}{\sum_{i=1}^{F}\mathrm{area}(T_i)}}.$$

4 Mesh parameterization

In this section, we describe our mesh parameterization technique (ARAP++). The input is a mesh of disk topology, and the output is a free-boundary flattened result. The algorithm consists of two main steps, local phase and global phase. Our method further extends and improves the ARAP approach (Liu *et al.*, 2008).

4.1 Overview of the ARAP++ approach

We outline the method in Algorithm 1, and the details are given in the following subsections.

4.2 Local phase

ARAP++ is also a linear iteration scheme. It requires an initial parameterization to start it off. We can perform the initialization with the shape-preserving method (Floater, 1997), as shown in Fig. 2. The local 1-ring patches of a 3D mesh S can be flattened to a plane using a method based on discrete exponential mapping. The 3D 1-ring patches of $\boldsymbol{x}_i$ (i.e., $N(\boldsymbol{x}_i)$) can be mapped to a 2D 1-ring

Algorithm 1 ARAP++

Input: A 3D mesh, S
Output: A 2D mesh, S^*
$S^* = S_0^*$; // Initial phase
loop
  // Local phase
  for $i = 1 : n$ **do**
    RingNodes($\boldsymbol{p}_i$) $\leftarrow$ LocalFlatten(RingNodes($\boldsymbol{x}_i$));
    $\triangle p_i p_j p_{j+1} \leftarrow$ RingPatches($\boldsymbol{p}_i$);
    $\triangle q_i q_j q_{j+1} \leftarrow$ RingPatches($\boldsymbol{q}_i$);
    $\omega_{i,j} \leftarrow$ ComputeWeights(RingNodes($\boldsymbol{x}_i$));
    $\boldsymbol{J}_{(i,j,j+1)} \leftarrow$ Jacobian($\triangle p_i p_j p_{j+1}$, $\triangle q_i q_j q_{j+1}$);
    $\boldsymbol{L}_{(i,j,j+1)} \leftarrow$ FittingMatrix($\boldsymbol{J}_{(i,j,j+1)}$);
  end for
  // Global phase
  $S_1^* \leftarrow$ ComputePara(ω, $\boldsymbol{L}$);
  $S_1^* \leftarrow$ PostProcess(S_1^*);
  if $d(S_1^* - S^*) > \varepsilon$ **then**
    $S_0^* = S_1^*$;
  else
    $S^* = S_1^*$;
    break;
  end if
end loop

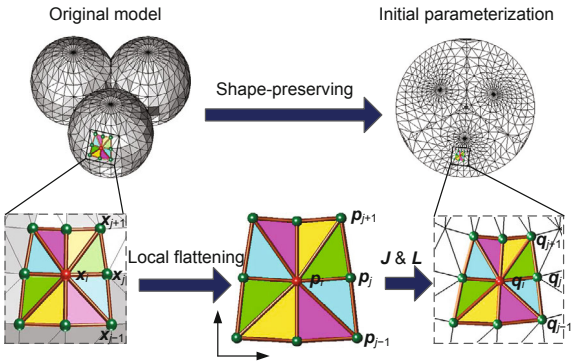

Fig. 2 Balls model: original model, initial parameterization, and local flattening

patches of $\boldsymbol{p}_i$ (i.e., $N(\boldsymbol{p}_i)$) in the plane, where the mapping should satisfy the following two equations:

$$\|\boldsymbol{p}_j - \boldsymbol{p}_i\| = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|,$$

$$\mathrm{ang}(\boldsymbol{p}_j, \boldsymbol{p}_i, \boldsymbol{p}_{j+1}) = \frac{2\pi\cdot\mathrm{ang}(\boldsymbol{x}_j, \boldsymbol{x}_i, \boldsymbol{x}_{j+1})}{\sum_{j\in N(i)}\mathrm{ang}(\boldsymbol{x}_j, \boldsymbol{x}_i, \boldsymbol{x}_{j+1})},$$

where $\mathrm{ang}(\boldsymbol{p}_j, \boldsymbol{p}_i, \boldsymbol{p}_{j+1})$ represents $\angle p_j p_i p_{j+1}$. Note that $N(\boldsymbol{q}_i)$ is the local 1-ring patches of the initial parameterization S^*. There exists an affine mapping f from $\triangle p_i p_j p_{j+1}$ to $\triangle q_i q_j q_{j+1}$. The Jacobian matrix $\boldsymbol{J}_{(i,j,j+1)} \in \mathbb{R}^{2\times 2}$ of f should

satisfy

$$\boldsymbol{q}_i - \boldsymbol{q}_j = \boldsymbol{J}_{(i,j,j+1)} \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j). \tag{4}$$

The local flattening of $N(\boldsymbol{p}_i)$ well preserves the shape (angle and area) of $N(\boldsymbol{x}_i)$. In the following process, we wish $N(\boldsymbol{q}_i)$ preserves the shape of $N(\boldsymbol{p}_i)$ as much as possible, so that it can preserve the shape of $N(\boldsymbol{x}_i)$ indirectly.

According to Eq. (4), if $N(\boldsymbol{q}_i)$ preserves the local structure of $N(\boldsymbol{p}_i)$, we need to search for a fitting matrix $\boldsymbol{L}_{(i,j,j+1)} \in \mathbb{R}^{2\times 2}$ to approximate $\boldsymbol{J}_{(i,j,j+1)}$, and it satisfies

$$\boldsymbol{q}_i - \boldsymbol{q}_j = \boldsymbol{L}_{(i,j,j+1)}(\boldsymbol{p}_i - \boldsymbol{p}_j),$$

where $\boldsymbol{L}_{(i,j,j+1)}$ belongs to a prescribed family of transformations (Gower and Dijksterhuis, 2004). The singular values of $\boldsymbol{L}$ (δ_1 and δ_2) should have some special properties, such as conformal ($\delta_1/\delta_2 = 1$), authalic ($\delta_1\delta_2 = 1$), or isometric ($\delta_1 = \delta_2 = 1$). The distance between $\boldsymbol{J}_{(i,j,j+1)}$ and $\boldsymbol{L}_{(i,j,j+1)}$ is measured by the Frobenius norm (Horn and Johnson, 1990). We employ the function $D(\boldsymbol{J}, \boldsymbol{L})$ to express the distance:

$$\begin{aligned} D(\boldsymbol{J}, \boldsymbol{L}) &= \left\| \boldsymbol{J}_{(i,j,j+1)} - \boldsymbol{L}_{(i,j,j+1)} \right\|_{\mathrm{F}}^2 \\ &= (\sigma_1 - \delta_1)^2 + (\sigma_2 - \delta_2)^2, \end{aligned}$$

where

$$\begin{cases} \boldsymbol{J}_{(i,j,j+1)} = \boldsymbol{U} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \boldsymbol{V}^{\mathrm{T}}, \\ \boldsymbol{L}_{(i,j,j+1)} = \boldsymbol{U} \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} \boldsymbol{V}^{\mathrm{T}}, \end{cases}$$

and σ_1, σ_2 are the singular values of $\boldsymbol{J}_{(i,j,j+1)}$.

From the properties of the Jacobian matrix (Floater and Hormann, 2005), we infer the following three conclusions (Fig. 3):

Conclusion 1 If $\delta_1 = \delta_2 = (\sigma_1 + \sigma_2)/2$, then $\boldsymbol{L}$ is a conformal matrix.

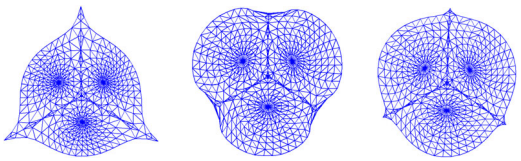

Fig. 3 Three cases of the fitting matrix for the Balls model: (a) conformal; (b) authalic; (c) isometric

Conclusion 2 If $\delta_1 = \delta$, $\delta_2 = 1/\delta$, then $\boldsymbol{L}$ is an authalic matrix, and δ is a real root of the following nonlinear equation:

$$\delta^4 - \sigma_1 \cdot \delta^3 + \sigma_2 \cdot \delta - 1 = 0.$$

The computation details are given in the Appendix.

Conclusion 3 If $\delta_1 = \delta_2 = 1$, then $\boldsymbol{L}$ is an isometric matrix.

Given a local map $f_i : f_i(\boldsymbol{p}_i) = \boldsymbol{q}_i$ in the 1-ring neighborhood of $\boldsymbol{x}_i$, based on the spring energy (Hoppe *et al.*, 1993) in Eq. (2), we define the deformation energy $E(f_i)$ from $N(\boldsymbol{p}_i)$ to $N(\boldsymbol{q}_i)$ as follows. When the 1-ring patches of $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ are oriented counterclockwise, we have

$$E(f_i) = \sum_{j\in N(i)} \omega_{i,j}^{(1)} \|(\boldsymbol{q}_i - \boldsymbol{q}_j) - \boldsymbol{L}_{(i,j,j+1)}(\boldsymbol{p}_i - \boldsymbol{p}_j)\|^2.$$

When the 1-ring patches of $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ are oriented clockwise, we have

$$E(f_i) = \sum_{j\in N(i)} \omega_{i,j}^{(2)} \|(\boldsymbol{q}_i - \boldsymbol{q}_j) - \boldsymbol{L}_{(i,j,j-1)}(\boldsymbol{p}_i - \boldsymbol{p}_j)\|^2.$$

In ARAP++, the 1-ring nodes have the same weights in both cases, that is, $\omega_{i,j} = \omega_{i,j}^{(1)} = \omega_{i,j}^{(2)}$. By adding these two equations, we can obtain the local deformation energy $E(f_i)$ as

$$\begin{aligned} E(f_i) = \sum_{j\in N(i)} \frac{\omega_{i,j}}{2} \Big(&\|(\boldsymbol{q}_i - \boldsymbol{q}_j) - \boldsymbol{L}_{(i,j,j+1)}(\boldsymbol{p}_i - \boldsymbol{p}_j)\|^2 \\ &+ \|(\boldsymbol{q}_i - \boldsymbol{q}_j) - \boldsymbol{L}_{(i,j,j-1)}(\boldsymbol{p}_i - \boldsymbol{p}_j)\|^2 \Big). \end{aligned} \tag{5}$$

According to the above derivation, we can recover the ARAP approach as a special case. Based on the Dirichlet energy (Pinkall and Polthier, 1993) over the whole 1-ring neighborhood in Eq. (3), when $\omega_{i,j}^{(1)} \neq \omega_{i,j}^{(2)}$, they are represented as

$$\begin{cases} \omega_{i,j}^{(1)} = \cot(\mathrm{ang}(\boldsymbol{x}_i, \boldsymbol{x}_{j+1}, \boldsymbol{x}_j)), \\ \omega_{i,j}^{(2)} = \cot(\mathrm{ang}(\boldsymbol{x}_i, \boldsymbol{x}_{j-1}, \boldsymbol{x}_j)). \end{cases}$$

Hence, the deformation energy $E(f_i)$ in Eq. (5) is the same as the local energy in the ARAP approach.

4.3 Global phase

After computing the local energy, we sum it to obtain the global energy:

$$W(S^*) = \sum_{i=1}^{n} E(f_i).$$

To optimize $W(S^*)$, we compute the gradient of $W(S^*)$ with respect to the positions $\boldsymbol{q}_i$, that is, $\partial W(S^*)/\partial \boldsymbol{q}_i = 0$. Then we have

$$\sum_{j\in N(i)} \omega_{i,j}(\boldsymbol{q}_i - \boldsymbol{q}_j) = \sum_{j\in N(i)} \frac{\omega_{i,j}}{2}\left(\boldsymbol{L}_{(i,j,j+1)} + \boldsymbol{L}_{(i,j,j-1)}\right)(\boldsymbol{p}_i - \boldsymbol{p}_j). \tag{6}$$

We can stitch together the flattened 1-ring patches according to Eq. (6), which is a global Poisson equation with two degrees of freedom ($\omega_{i,j}$ and $\boldsymbol{L}$). To achieve the best possible results, several iterations should be performed while updating $\boldsymbol{L}$ and $\boldsymbol{p}$ of the right-hand items after each iteration. As shown in Fig. 4, compared with the ARAP approach, ARAP++ can obtain the flattened result by stitching together the 1-ring patches, instead of individual triangles.

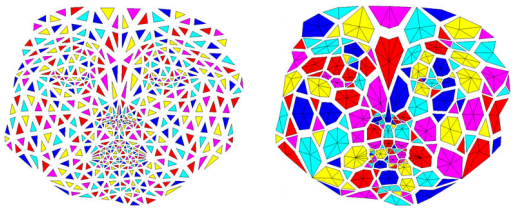

Fig. 4 Flattened results by stitching the individual triangles under ARAP (a) and 1-ring patches under ARAP++ (b) for the Face model

Note that the coefficient matrix $\boldsymbol{T} = [\omega_{i,j}]_{n\times n}$ is a singular matrix. Thus, we should select a node fixed on the plane before iterations, so that $\boldsymbol{T}$ turns to a non-singular matrix. In our algorithm, the fixed node in the plane is selected arbitrarily. Generally, we choose a node in the relative flat region and far away from the boundary. Unfortunately, sometimes large distortion and overlapping will still be produced around the fixed point. In this case, we should do some post-processing operations to cope with this problem. If the fixed node is located outside its 1-ring polygon (Fig. 5a), then there must be overlapping, and we should relocate it in the center of the 1-ring polygon. As shown in Fig. 5b, this represents a simple and efficient way to deal with this problem. If the overlapping of many nodes still occurs, we choose another fixed node to start the computation and restart the iteration.

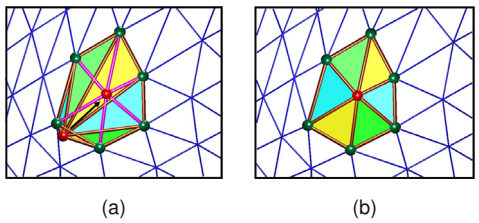

Fig. 5 Post-processing of the fixed point: (a) before processing; (b) after processing

Our method has the same local/global procedure as the ARAP approach. So, it is guaranteed to converge to a local minimum. Empirically, our method appears to converge with the increase of the number of iterations. As shown in Fig. 6, the global energy always decreases with each iteration, and finally stabilizes at a minimum. Experimental results also demonstrate that the distortions (angle and area) are decreasing progressively with the number of iterations (as shown later in Table 2). As the termination criterion of our algorithm, we require that the distortions (angle and area) between the previous and next iterations should be under the threshold 10^{-3}. We also can set a maximum number of iterations. Numerical results suggest that the maximum number of iterations is less than five.

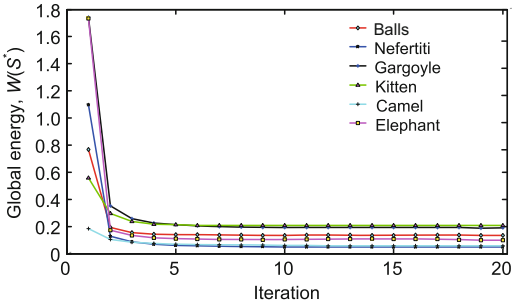

Fig. 6 Numerical tests for convergence

4.4 Boundary weights

Our method needs various different computations connecting between internal weights and boundary weights. The internal weights are determined by the convex combination of their 1-ring nodes' weights. However, we still have to perform some necessary preliminary operations to determine the boundary weights:

1. Uniform weight

The weight of $\boldsymbol{x}_j$ is $\omega_{i,j} = 1/d_i$.

2. Shape-preserving weight

Using an isometric mapping to locate $\boldsymbol{x}_i$ and its 1-ring nodes $\boldsymbol{x}_j$, $\boldsymbol{x}_{n_j}$, $\boldsymbol{x}_{n_k}$, $\boldsymbol{x}_k$ on the plane, we obtain $\boldsymbol{p}_i$ and the 1-ring nodes $\boldsymbol{p}_j$, $\boldsymbol{p}_{n_j}$, $\boldsymbol{p}_{n_k}$, $\boldsymbol{p}_k$ (Fig. 7). To generate a local 1-ring neighborhood, two virtual boundary nodes $\boldsymbol{p}_{n_j}$, $\boldsymbol{p}_{n_k}$ are inserted between $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$. The inserted nodes should satisfy the following equations:

$$\begin{cases} \mathrm{ang}(\boldsymbol{p}_j,\boldsymbol{p}_i,\boldsymbol{p}_{m_j}) = (2\pi - \mathrm{ang}(\boldsymbol{p}_j,\boldsymbol{p}_i,\boldsymbol{p}_k))/3, \\ \mathrm{ang}(\boldsymbol{p}_{m_j},\boldsymbol{p}_i,\boldsymbol{p}_{m_k}) = (2\pi - \mathrm{ang}(\boldsymbol{p}_j,\boldsymbol{p}_i,\boldsymbol{p}_k))/3, \\ \mathrm{ang}(\boldsymbol{p}_{m_k},\boldsymbol{p}_i,\boldsymbol{p}_k) = (2\pi - \mathrm{ang}(\boldsymbol{p}_j,\boldsymbol{p}_i,\boldsymbol{p}_k))/3, \\ \|\boldsymbol{p}_i - \boldsymbol{p}_{m_j}\| = \|\boldsymbol{p}_i - \boldsymbol{p}_{m_k}\| = \dfrac{\sum_{j\in N(i)} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|}{d_i}. \end{cases}$$

Then the weights $\omega_{i,j}$ of $\boldsymbol{p}_j$, $\boldsymbol{p}_{n_j}$, $\boldsymbol{p}_{n_k}$, and $\boldsymbol{p}_k$ can be computed in the local 1-ring neighborhood.

3. Mean-value, cotan, and intrinsic weights

Fig. 7 shows that the 1-ring nodes $\boldsymbol{x}_j$, $\boldsymbol{x}_{n_j}$, $\boldsymbol{x}_{n_k}$, $\boldsymbol{x}_k$ can be divided into two parts: $\boldsymbol{x}_{n_j}$, $\boldsymbol{x}_{n_k}$ (intermediate nodes), and $\boldsymbol{x}_j$, $\boldsymbol{x}_k$ (endpoints). The weights of $\boldsymbol{x}_{n_j}$, $\boldsymbol{x}_{n_k}$ are consistent with those of the internal nodes. However, the weights of $\boldsymbol{x}_j$, $\boldsymbol{x}_k$ are computed in $\triangle x_i x_j x_{n_j}$ and $\triangle x_i x_k x_{n_k}$, respectively.

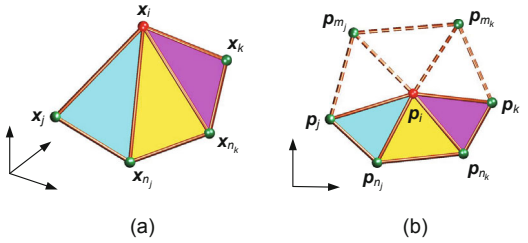

Fig. 7 The 3D boundary 1-ring nodes (a) and the associated flattened version (b)

4.5 Multi-boundary flattening

There are a couple of ways to flatten a multi-boundary mesh by using ARAP++. In the first approach, the mesh is initialized by flattening using the LSCM method (Lévy *et al.*, 2002). Then successive iterations of Eq. (6) are applied in order to achieve the final flattening result. However, if the inner boundaries (holes) are located at the high-curvature areas, then overlapping will be produced at these boundaries. In this study, we employ the second approach as follows:

Step 1: Add a virtual node to the center of the holes, connecting the virtual nodes with the 1-ring nodes, and thus the multi-boundary mesh turns into a single-boundary mesh.

Step 2: Compute the flattened result according to Algorithm 1.

Step 3: Remove the virtual nodes and connections from the holes.

It can be seen from Fig. 8 that the first approach generates overlapping around the holes, while the second one apparently produces better results than the first one. If the above virtual node method still induces considerable distortion, there are a number of quite advanced and difficult methods for filling the holes, such as Delaunay triangulation (Lawson, 1977).

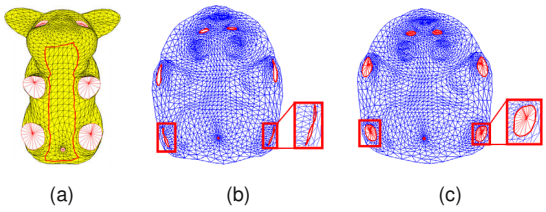

Fig. 8 Flattening of a multi-boundary mesh for the Pig model: (a) original model; (b) the first approach (without filled holes); (c) the second approach (with filled holes)

4.6 Stretch operator

To reduce the area and stretch distortions for high-curvature models, the stretch operator (Sander *et al.*, 2001; Yoshizawa *et al.*, 2004) is employed to improve our scheme (Eq. (6)). In addition, it can make ARAP++ reduce flipping and overlapping during parameterization.

As shown in Fig. 2, the Jacobian matrix $\boldsymbol{J}_{(i,j,j+1)}$ between $\triangle p_i p_j p_{j+1}$ and $\triangle q_i q_j q_{j+1}$ can be obtained according to Eq. (4). The singular values of $\boldsymbol{J}_{(i,j,j+1)}$ are σ_1 and σ_2. The stretch operator of $\boldsymbol{x}_i$ is defined as

$$\mu_i = \sqrt{\frac{\sum_{j\in N(i)} \mathrm{area}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_{j+1}) \cdot \sigma^2_{(i,j,j+1)}}{\sum_{j\in N(i)} \mathrm{area}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_{j+1})}},$$

where $\sigma_{(i,j,j+1)} = \sqrt{(\sigma_1^2 + \sigma_2^2)/2}$.

In essence, the stretch operator can change convex combination weights proportionally to the area of the 1-ring patches. We can improve Eq. (6) by

employing the following formula:

$$\sum_{j\in N(i)} (\mu_i)^{\theta}\omega_{i,j}(\boldsymbol{q}_i-\boldsymbol{q}_j) = \sum_{j\in N(i)} \frac{\omega_{i,j}}{2}\left(\boldsymbol{L}_{(i,j,j+1)}+\boldsymbol{L}_{(i,j,j-1)}\right)(\boldsymbol{p}_i-\boldsymbol{p}_j). \tag{7}$$

To achieve the valid results, Eq. (7) requires only one step of iteration in computation. If the number of iterations increases, there will be overlapping in the final results. To control stretch distortion, the exponent θ should be adjusted according to different models. Fig. 9 shows the results of parameterization with different θ, where $\omega_{i,j}$ represents the mean-value weight, and the singular values of $\boldsymbol{L}$ are $\delta_1=\delta_2=(\sigma_1+\sigma_2)/2$.

5 Simulations and comparisons

All the experiments were tested under MATLAB on a Pentium® Dual-Core, 2.5 GHz CPU computer with 4 GB RAM. To confirm the effectiveness of our method, we carried out simulations on several typical models.

5.1 Comparison of five classic weights

In this subsection, we compare the flattening results for five different types of weights ($\omega_{i,j}$ in Eq. (6)), namely uniform, shape-preserving, mean-value, cotan, and intrinsic. The singular values of the fitting matrix $\boldsymbol{L}$ are $\delta_1=\delta_2=1$. As shown in Fig. 10, two models (Balls, Nefertiti) have been tested. For comparison, the final results are displayed for the same initial value and three iterations. Table 1 illustrates the angle and area distortions corresponding to the five classic types of weights, showing that

Table 1 Comparison of distortion measures of five classic types of weights on two standard test models

Method	Balls[a]		Nefertiti[b]	
	Angle distortion	Area distortion	Angle distortion	Area distortion
Uniform	0.195	0.233	0.157	0.213
Shape-preserving	0.180	0.201	0.125	0.173
Mean-value	**0.117**	0.187	**0.116**	0.167
Cotan	0.124	**0.184**	0.118	**0.158**
Intrinsic	0.177	0.211	0.129	0.195

[a] Number of vertices: 547, number of faces: 1032; [b] Number of vertices: 661, number of faces: 1252

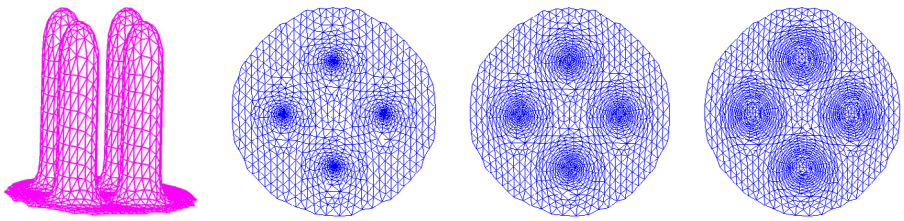

Fig. 9 The 3D mesh (a) and the stretch operator for $\theta=0$ (b), $\theta=1$ (c), and $\theta=1.6$ (d) for the Bump model

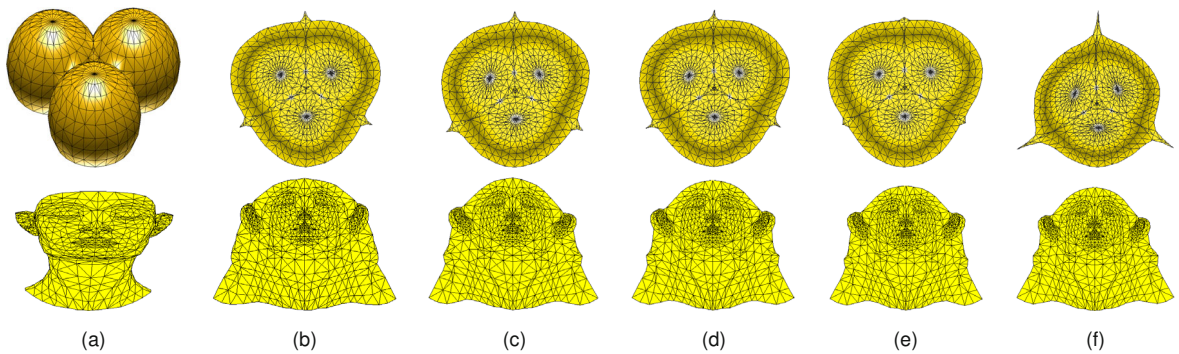

Fig. 10 Original models (a) and application results of ARAP++ for the uniform (b), shape-preserving (c), mean-value (d), cotan (e), and intrinsic (f) weights. The first line: Balls model; the second line: Nefertiti model

the mean-value weight and cotan weight perform better than most of the others in angle distortion. Moreover, the area distortion of the mean-value and cotan weights are even smaller than others. However, the cotan weight produces instability, and also negative values in some cases. In summary, we can ascertain that the mean-value weight proves the best choice in the following simulations.

5.2 Comparison of ARAP and ARAP++

In this subsection, ARAP++ is chiefly inspired by ARAP parameterization (Liu *et al.*, 2008). We improve the local phase which is based on discrete exponential maps of the 1-ring patches instead of flattening of individual triangles, as in the original ARAP method. We compare ARAP++ with ARAP in three iterations, where $\omega_{i,j}$ represents the mean-value weight, and the singular values of $\boldsymbol{L}$ are $\delta_1 = \delta_2 = 1$. Two models (Gargoyle, Triceratops) have been tested via two methods. As shown in Fig. 11, we compare the parameterizations and texture mappings between ARAP and ARAP++. Table 2 illustrates the angle distortion, area distortion, and running time of two methods during iterations 1–3. The two methods render the same level in angle distortion, and ARAP++ slightly performs better than ARAP in area distortion. However, ARAP++ has a higher computational complexity, so it is more time-consuming compared with ARAP.

In Fig. 12, the ARAP method is shown to suffer

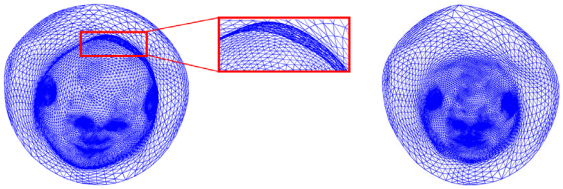

Fig. 12 Parameterization of the high-curvature Mannequin model: (a) a detailed overlap under ARAP; (b) a valid parameterization result under ARAP++

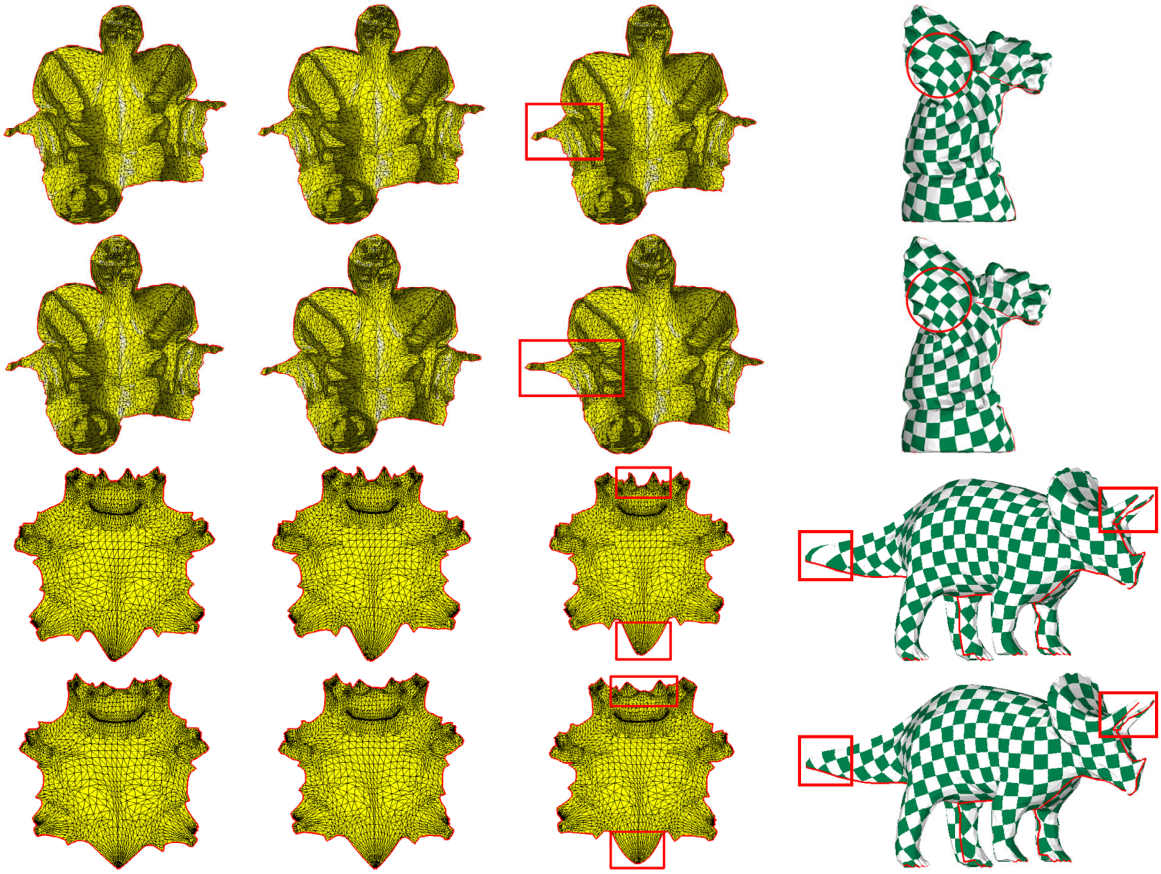

Fig. 11 Parameterizations and textures of ARAP (the first and third lines) and ARAP++ (the second and fourth lines)

from overlapping when dealing with high-curvature models. In contrast, ARAP++ can obtain valid flattening results by choosing an appropriate exponent value ($\theta = -0.2$) provided by Eq. (7). Therefore, ARAP++ outperforms ARAP in terms of controlling stretch distortion.

5.3 Comparison with several state-of-the-art methods

In this subsection, we carry out three simulations to compare ARAP++ with several state-of-the-art parameterization methods, namely LSCM, LABF, and BD-ARAP (conformal distortion $C = 2$).

In the first simulation, ARAP++ employs Eq. (6), using three iterations, where $\omega_{i,j}$ represents the mean-value weight, and the singular values of $\boldsymbol{L}$ are $\delta_1 = \delta_2 = 1$. As shown in Fig. 13, the single-boundary models (Hand, Camel) and multi-boundary models (Kitten, Elephant) are cut along the red line. We compare the parameterizations and textures rendered by these methods. Table 3 illustrates the angle distortion, area distortion, and running time of the four methods. It reveals that ARAP++ produces a slightly larger angle distortion than the other three methods, but it renders the best result in area distortion. In other words, ARAP++ provides a good trade-off between the overall angle and area distortions, which leads to good results in texture mapping. Regarding the comparison of running time, ARAP++ is shown to be faster than LABF and BD-ARAP. However, due to the necessity of several iterations in the computation of ARAP++, it is slower than LSCM.

In the second simulation, we compare the

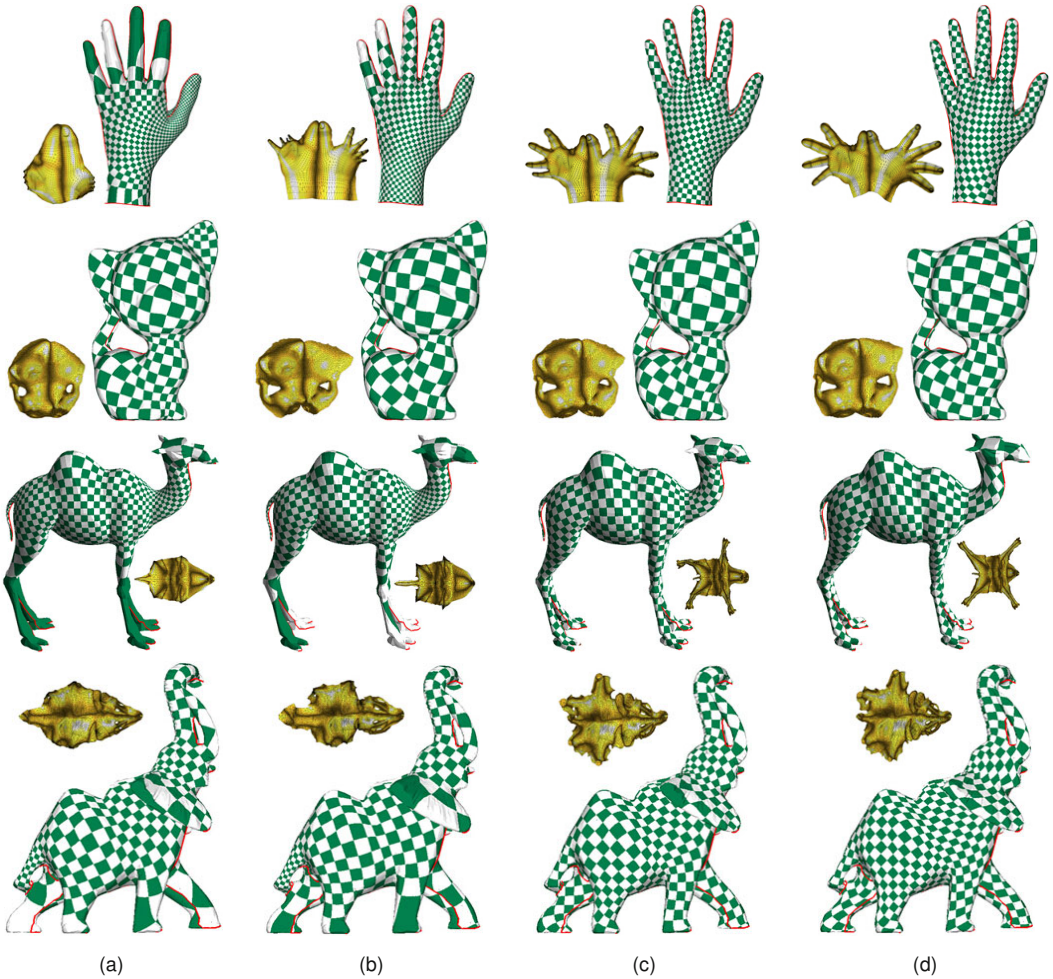

Fig. 13 Parameterizations and textures: (a) LSCM; (b) LABF; (c) BD-ARAP; (d) ARAP++. References to color refer to the online version of this figure

parameterizations and textures of high-curvature models (Mushroom, Mannequin, Lion) which have not been cut in advance. ARAP++ employs Eq. (7), where $\omega_{i,j}$ is chosen to be the mean-value weight, and the singular values of $\boldsymbol{L}$ are $\delta_1 = \delta_2 = 1$. The exponent θ can be adjusted to obtain the valid results. When performing the texture mapping of high-curvature models, we should have a quadrilateral remeshing of the original model according to the parameterization. The details of this remeshing method can be found in Hormann and Greiner (2000b). In contrast, the remeshed models are given by the corresponding sets of vertices, facets, and topological connection. During the uniform subdivision of the base mesh, we apply the same number of umbrella operators (Hormann and Greiner, 2000b) on the remeshed models. Table 4 illustrates the angle, area, and strech (L^2) distortions of the four methods. ARAP++ shows better area and stretch distortions than others. However, ARAP++ leads to the highest angle distortion. As shown in Fig. 14, the textured models produced by ARAP++ are more uniform and aesthetical than others.

In the third simulation, a remeshing of the triangulation to the original model is performed. The details of the remeshing method can be found in Hormann *et al.* (2001). Again, the remeshed models are given by the corresponding sets of vertices, edges, and facets as above. Table 5 illustrates the maximum and minimum facet areas of the models. It reveals that the maximum facet area is smaller and the minimum facet area is larger than those of other methods. As shown in Fig. 15, the remeshing results of ARAP++ outperform others in terms of facets' uniformity.

6 Conclusions and outlook

Even though mesh parameterization has received considerable attention in computer graphics for more than two decades, there are still many complicated problems that need to be resolved. In particular, there still does not exist definite solutions regarding the optimal trade-off between angle and area distortions during the parameterization process. In this paper, we have presented a simple and efficient approach, ARAP++, to flatten a 3D mesh surface. This is an extension of the local/global approach ARAP, which can obtain the flattened results by stitching together the 1-ring patches. In addition, to deal with high-curvature models, the stretch operator is introduced, which can better control area and stretch distortions, and has attained better visualization performance in

Table 2 Comparison of distortion measures and time of ARAP and ARAP++

Method	Iteration index	Gargoyle[a]			Triceratops[b]		
		Angle distortion	Area distortion	Time (s)	Angle distortion	Area distortion	Time (s)
ARAP	1	**0.147**	0.244	**9.05**	0.177	0.164	**9.44**
	2	**0.140**	0.234	**13.39**	0.147	0.148	**14.01**
	3	**0.137**	0.227	**18.47**	0.134	0.141	**18.87**
ARAP++	1	0.168	**0.217**	9.59	**0.156**	**0.158**	10.74
	2	0.150	**0.203**	14.06	**0.129**	**0.147**	14.93
	3	0.141	**0.196**	18.73	**0.116**	**0.138**	19.43

[a] Number of vertices: 2607, number of faces: 5000; [b] Number of vertices: 3015, number of faces: 5660

Table 3 Comparison of three parameterization methods and ARAP++ on four standard test meshes

Method	Hand[a]			Kitten[b]			Camel[c]			Elephant[d]		
	Angle distortion	Area distortion	Time (s)	Angle distortion	Area distortion	Time (s)	Angle distortion	Area distortion	Time (s)	Angle distortion	Area distortion	Time (s)
LSCM	**0.009**	0.991	**8.74**	0.044	0.416	**3.79**	0.099	0.731	**3.59**	0.063	0.699	**12.67**
LABF	0.015	0.659	35.05	**0.026**	0.359	16.70	**0.053**	0.703	14.79	**0.047**	0.715	45.90
BD-ARAP	0.019	0.269	80.78	0.039	0.273	68.78	0.064	0.254	141.12	0.065	0.380	239.23
ARAP++	0.065	**0.107**	15.80	0.102	**0.161**	10.14	0.137	**0.227**	10.05	0.166	**0.276**	25.25

[a] Number of vertices: 4911, number of faces: 9174; [b] Number of vertices: 3101, number of faces: 6000; [c] Number of vertices: 3101, number of faces: 5860; [d] Number of vertices: 5525, number of faces: 10 594

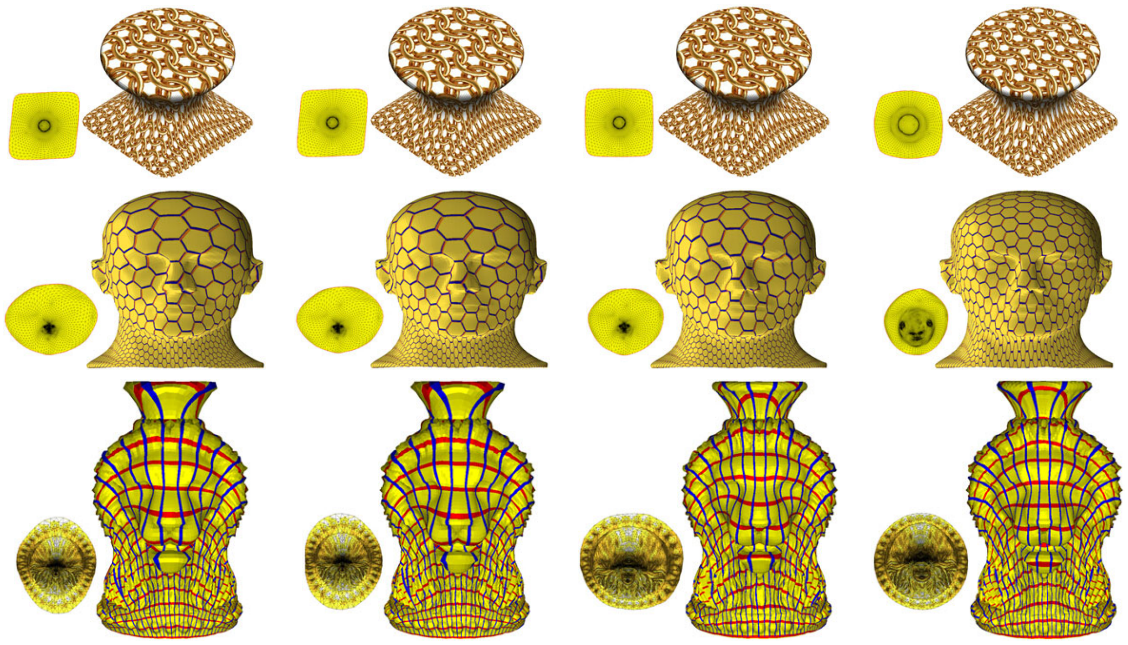

Fig. 14 Parameterizations and textures of high-curvature models: (a) LSCM; (b) LABF; (c) BD-ARAP; (d) ARAP++

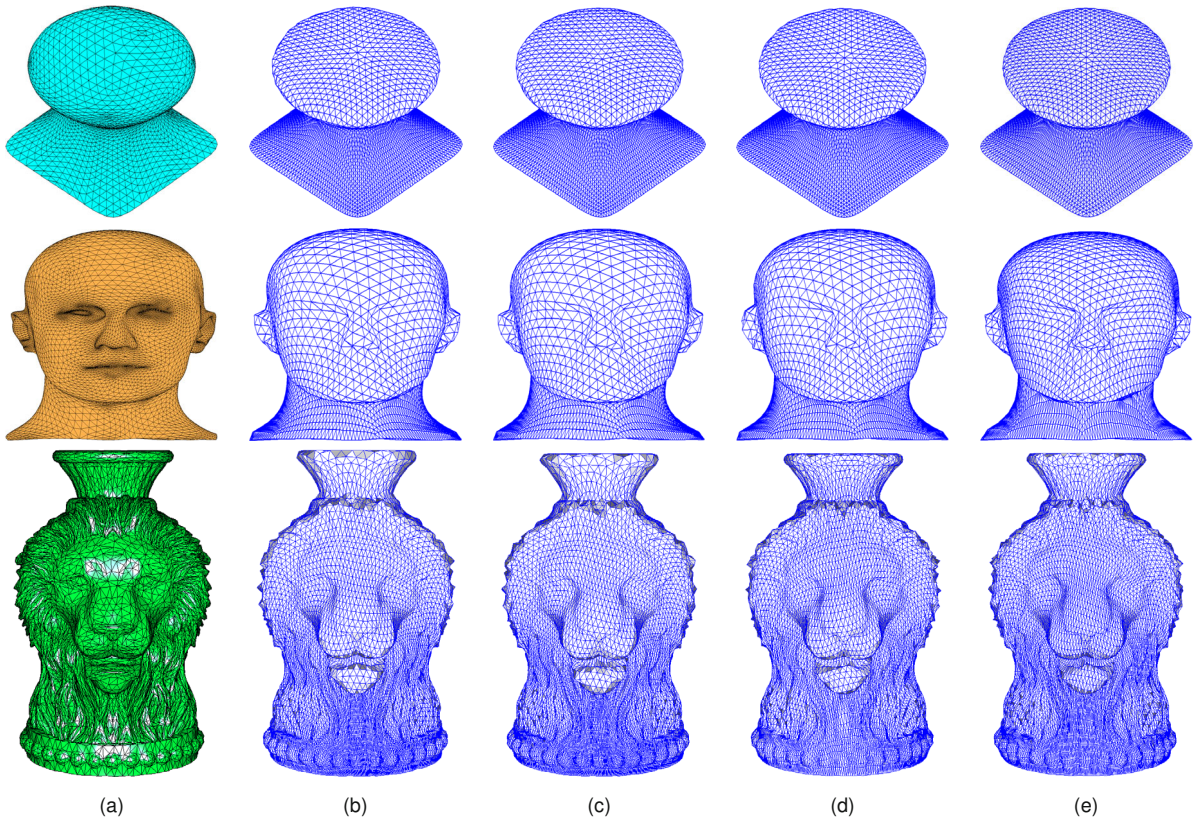

Fig. 15 Surface remeshing: (a) original mesh; (b) LSCM; (c) LABF; (d) BD-ARAP; (e) ARAP++

Table 4 Comparison of three parameterization methods and ARAP++ for high-curvature models

Method	Mushroom[a]			Mannequin[b]			Lion[c]		
	Angle distortion	Area distortion	Stretch distortion	Angle distortion	Area distortion	Stretch distortion	Angle distortion	Area distortion	Stretch distortion
LSCM	**0.026**	1.034	2.604	**0.027**	1.436	7.436	**0.071**	1.241	18.806
LABF	0.027	1.033	2.601	0.028	1.435	7.438	0.073	1.246	11.131
BD-ARAP	0.028	1.031	2.565	0.124	1.417	5.719	0.166	1.059	3.282
ARAP++	0.154	**0.807**	**1.535**	0.322	**0.887**	**1.654**	0.285	**0.866**	**2.883**

[a] Number of vertices: 2337, number of faces: 4608; [b] Number of vertices: 6743, number of faces: 13 424; [c] Number of vertices: 7218, number of faces: 14 371

Table 5 Minimum and maximum facet areas of the remeshed models for three standard test meshes

Method	Mushroom[a]		Mannequin[b]		Lion[c]	
	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
LSCM	2.746×10^{-4}	2.661×10^{-3}	1.345×10^{-4}	2.343×10^{-3}	3.928×10^{-5}	1.807×10^{-3}
LABF	2.028×10^{-4}	2.421×10^{-3}	1.345×10^{-4}	2.347×10^{-3}	3.576×10^{-5}	1.866×10^{-3}
BD-ARAP	3.131×10^{-4}	2.654×10^{-3}	1.536×10^{-4}	2.048×10^{-3}	4.159×10^{-5}	0.892×10^{-3}
ARAP++	$\mathbf{4.229}\times10^{-4}$	$\mathbf{1.911}\times10^{-3}$	$\mathbf{2.074}\times10^{-4}$	$\mathbf{1.229}\times10^{-3}$	$\mathbf{7.068}\times10^{-5}$	$\mathbf{0.635}\times10^{-3}$

[a] Number of vertices: 3681, number of faces: 7168; [b] Number of vertices: 3169, number of faces: 6144; [c] Number of vertices: 12 481, number of faces: 24 576

applications, such as texture mapping and surface remeshing.

Our future work will focus on extending the present method to mesh deformation and spherical parameterization, which will undoubtedly shed some new light on the improvement of computer graphics technology. Another natural direction would be to extend our results to larger patches (not just 1-rings), such as those produced by the flattening algorithm introduced in Saucan *et al.* (2008).

Acknowledgements

The authors would like to thank Dr. Zhao-liang MENG, Dr. Xin FAN, and Dr. Wan-feng QI for their constructive recommendations for this work.

References

Aigerman, N., Lipman, Y., 2013. Injective and bounded distortion mappings in 3D. *ACM Trans. Graph.*, **32**(4), Article 106. http://dx.doi.org/10.1145/2461912.2461931

Bouaziz, S., Deuss, M., Schwartzburg, Y., *et al.*, 2012. Shape-up: shaping discrete geometry with projections. *Comput. Graph. Forum*, **31**(5):1657-1667. http://dx.doi.org/10.1111/j.1467-8659.2012.03171.x

Chen, B.M., Gotsman, C., Bunin, G., 2008. Conformal flattening by curvature prescription and metric scaling. *Comput. Graph. Forum*, **27**(2):449-458. http://dx.doi.org/10.1111/j.1467-8659.2008.01142.x

Chen, Z., Liu, L., Zhang, Z., *et al.*, 2007. Surface parameterization via aligning optimal local flattening. Proc. Symp. on Solid and Physical Modeling, p.291-296. http://dx.doi.org/10.1145/1236246.1236287

Degener, P., Meseth, J., Klein, R., 2003. An adaptable surface parameterization method. Proc. 12th Int. Meshing Roundtable, p.227-237.

Desbrun, M., Meyer, M., Allize, P., 2002. Intrinsic parameterization of surface meshes. *Comput. Graph. Forum*, **21**(2):209-218. http://dx.doi.org/10.1111/1467-8659.00580

Eck, M., DeRose, T., Duchamp, T., *et al.*, 1995. Multiresolution analysis of arbitrary meshes. Proc. 22nd Annual Conf. on Computer Graphics and Interactive Techniques, p.173-182. http://dx.doi.org/10.1145/218380.218440

Floater, M.S., 1997. Parameterization and smooth approximation of surface triangulations. *Comput. Aid. Geom. Des.*, **14**(3):231-250. http://dx.doi.org/10.1016/S0167-8396(96)00031-3

Floater, M.S., 2003. Mean value coordinates. *Comput. Aid. Geom. Des.*, **20**(1):19-27. http://dx.doi.org/10.1016/S0167-8396(03)00002-5

Floater, M.S., Hormann, K., 2005. Surface parameterization: a tutorial and survey. *In*: Dodgson, N.A., Floater, M.S., Sabin, M.A. (Eds.), Advances in Multiresolution for Geometric Modelling, p.157-186. http://dx.doi.org/10.1007/3-540-26808-1_9

Gortler, S., Gotsman, C., Thurston, D., 2006. Discrete one-forms on meshes and applications to 3D mesh parameterization. *Comput. Aid. Geom. Des.*, **23**(2):83-112. http://dx.doi.org/10.1016/j.cagd.2005.05.002

Gower, J.C., Dijksterhuis, G.B., 2004. Procrustes Problems. Oxford University Press, Oxford.

Gu, X., Yau, S., 2002. Computing conformal structures of surfaces. *Commun. Inform. Syst.*, **2**(2):121-146. http://dx.doi.org/10.4310/CIS.2002.v2.n2.a2

Gu, X., Yau, S., 2003. Global conformal surface parameterization. Proc. Eurographics/ACM SIGGRAPH Symp. on Geometry Processing, p.127-137.

Haker, S., Angenent, S., Tannenbaum, A., *et al.*, 2000. Conformal surface parameterization for texture mapping. *IEEE Trans. Visual. Comput. Graph.*, **6**(2):181-189. http://dx.doi.org/10.1109/2945.856998

Hoppe, H., DeRose, T., Duchamp, T., *et al.*, 1993. Mesh optimization. Proc. 20th Annual Conf. on Computer Graphics and Interactive Techniques, p.19-26. http://dx.doi.org/10.1145/166117.166119

Hormann, K., Greiner, G., 2000a. MIPS: an efficient global parameterization method. Proc. Curve and Surface, p.153-162.

Hormann, K., Greiner, G., 2000b. Quadrilateral remeshing. Proc. Vision Modeling and Visualization, p.153-162.

Hormann, K., Greiner, G., Campagna, S., 1999. Hierarchical parameterization of triangulated surfaces. Proc. of Vision, Modeling and Visualization, p.219-226.

Hormann, K., Labsik, U., Greiner, G., 2001. Remeshing triangulated surfaces with optimal parameterizations. *Comput.-Aid. Des.*, **33**(11):779-788. http://dx.doi.org/10.1016/S0010-4485(01)00094-X

Hormann, K., Lévy, B., Sheffer, A., 2007. Mesh parameterization: theory and practice. Proc. SIGGRAPH, p.1-122.

Horn, R., Johnson, C., 1990. Norms for vectors and matrices. *In*: Matrix Analysis. Cambridge University Press, England.

Jacobson, A., Baran, I., Kavan, L., *et al.*, 2012. Fast automatic skinning transformations. *ACM Trans. Graph.*, **31**(4), Article 77. http://dx.doi.org/10.1145/2185520.2185573

Jin, M., Kim, J., Luo, F., *et al.*, 2008. Discrete surface Ricci flow. *IEEE Trans. Visual. Comput. Graph.*, **14**(5):1030-1043. http://dx.doi.org/10.1109/TVCG.2008.57

Kharevych, L., Springborn, B., Schröder, P., 2006. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, **25**(2):412-438. http://dx.doi.org/10.1145/1138450.1138461

Lawson, L., 1977. Software for c1 surface interpolation. *In*: Mathematical Software III. Academic Press, New York.

Lee, Y., Kim, H., Lee, S., 2002. Mesh parameterization with a virtual boundary. *Comput. Graph.*, **26**(5):677-686. http://dx.doi.org/10.1016/S0097-8493(02)00123-1

Levi, Z., Zorin, D., 2014. Strict minimizers for geometric optimization. *ACM Trans. Graph.*, **33**(6), Article 185. http://dx.doi.org/10.1145/2661229.2661258

Lévy, B., Petitjean, S., Ray, N., *et al.*, 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, **21**(3):362-371. http://dx.doi.org/10.1145/566570.566590

Lipman, Y., 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, **31**(4), Article 108. http://dx.doi.org/10.1145/2185520.2185604

Liu, L., Zhang, L., Xu, Y., *et al.*, 2008. A local/global approach to mesh parameterization. *Comput. Graph. Forum*, **27**(5):1495-1504. http://dx.doi.org/10.1111/j.1467-8659.2008.01290.x

Mullen, P., Tong, Y., Alliez, P., *et al.*, 2008. Spectral conformal parameterization. *Comput. Graph. Forum*, **27**(5):1487-1494. http://dx.doi.org/10.1111/j.1467-8659.2008.01289.x

Pinkall, U., Polthier, K., 1993. Computing discrete minimal surface and their conjugates. *Exp. Math.*, **2**(1):15-36. http://dx.doi.org/10.1080/10586458.1993.10504266

Sander, P., Snyder, J., Gortler, S., *et al.*, 2001. Texture mapping progressive meshes. Proc. 28th Annual Conf. on Computer Graphics and Interactive Techniques, p.409-416. http://dx.doi.org/10.1145/383259.383307

Saucan, E., Appleboim, E., Barak-Shimron, E., *et al.*, 2008. Local versus global in quasiconformal mapping for medical imaging. *J. Math. Imag. Vis.*, **32**(3):293-311. http://dx.doi.org/10.1007/s10851-008-0101-6

Sheffer, A., de Sturler, E., 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Eng. Comput.*, **17**(3):326-337. http://dx.doi.org/10.1007/PL00013391

Sheffer, A., Lévy, B., Mogilnitsky, M., *et al.*, 2005. ABF++: fast and robust angle based flattening. *ACM Trans. Graph.*, **24**(2):311-330. http://dx.doi.org/10.1145/1061347.1061354

Sheffer, A., Praun, E., Rose, K., 2007. Mesh parameterization methods and their applications. *Comput. Graph. Vis.*, **2**(2):105-171. http://dx.doi.org/10.1561/0600000011

Sorkine, O., Alexa, M., 2007. As-rigid-as-possible surface modeling. Proc. Eurographics Symp. on Geometry Processing, p.109-116.

Tutte, W.T., 1963. How to draw a graph. *Proc. London Math. Soc.*, **13**(3):743-768..

Weber, O., Zorin, D., 2014. Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph.*, **33**(4), Article 75. http://dx.doi.org/10.1145/2601097.2601227

Weber, O., Myles, A., Zorin, D., 2012. Computing extremal quasiconformal maps. *Comput. Graph. Forum*, **31**(5):1679-1689. http://dx.doi.org/10.1111/j.1467-8659.2012.03173.x

Yoshizawa, S., Belyaev, A., Seidel, H., 2004. A fast and simple stretch-minimizing mesh parameterization. Proc. Shape Modeling Applications, p.200-208. http://dx.doi.org/10.1109/SMI.2004.1314507

Zayer, R., Lévy, B., Seidel, H., 2007. Linear angle based parameterization. Proc. 5th Eurographics Symp. on Geometry Processing, p.135-141.

Zhang, L., Liu, L., Gotsman, C., *et al.*, 2010. Mesh reconstruction by meshless denoising and parameterization. *Comput. Graph.*, **34**(3):198-208. http://dx.doi.org/10.1016/j.cag.2010.03.006

Zhao, X., Su, Z., Gu, X., *et al.*, 2013. Area-preservation mapping using optimal mass transport. *IEEE Trans. Visual. Comput. Graph.*, **19**(12):2838-2847. http://dx.doi.org/10.1109/TVCG.2013.135

Zigelman, G., Kimmel, R., Kiryati, N., 2002. Texture mapping using surface flattening via multidimensional scaling. *IEEE Trans. Visual. Comput. Graph.*, **8**(2):198-207. http://dx.doi.org/10.1109/2945.998671

Appendix: Authalic fitting matrix

Let $\delta_1=\delta$ and $\delta_2=1/\delta$. The target function $E(\delta)$ can be expressed as follows:

$$E(\delta)=(\sigma_1-\delta)^2+(\sigma_2-1/\delta)^2.$$

To achieve the minimum $E(\delta)$, we compute the derivative of $E(\delta)$ with respect to δ, and arrive at

$$\frac{\partial E(\delta)}{\partial\delta}=\delta^4-\sigma_1\delta^3+\sigma_2\delta-1=0.$$

This is a nonlinear equation, and one of its real roots is the intersection point of the two curves y_1 and y_2 in Fig. A1:

$$\begin{cases} y_1=\delta^4-\sigma_1\delta^3, \\ y_2=1-\sigma_2\delta, \end{cases}$$

where $\sigma_1>0$, $\sigma_2>0$.

The interval of δ is $[\sigma_1,1/\sigma_2]$ or $[1/\sigma_2,\sigma_1]$. Then a numerical solution of δ can be obtained using the dichotomy method.

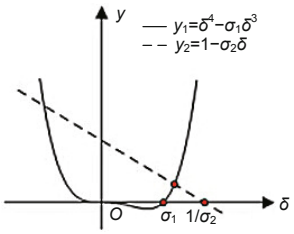

Fig. A1 Intersection of the two curves y_1 and y_2