第 50 卷第 12 期
2020 年 12 月
中 国 科 学 技 术 大 学 学 报
JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA
Vol. 50, No. 12
Dec. 2020
Received: 2020-07-16; Revised: 2020-09-06
doi:10.3969/j.issn.0253-2778.2020.12.003

Anisotropic surface meshing using locally isometric embedding

LI Huicong, FU Xiaoming*
School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China
* Corresponding author. E-mail: fuxm@ustc.edu.cn

Abstract: A novel method for anisotropic surface meshing was proposed. Different from the previous methods using globally conformal embeddings or high-dimensional isometric embeddings, our algorithm is based on the idea of locally isometric embedding. In order to achieve isometric embeddings, the input surface was partitioned into a set of cone patches that are remeshed one by one. First, a patch was parameterized bijectively into a plane, then an anisotropic mesh was generated in the parameterized domain, and finally, the remeshed patch was mapped back to the input surface. To deal with the stitching problem between different patches, the cone patch was made containing the previously unprocessed boundary. Therefore, the triangles near the boundary could be remeshed. The robustness of our method was demonstrated on various complex meshes. Compared to the existing methods, our method is more robust, and contains a smaller approximation error to the input mesh.

Keywords: Riemannian metric; cone patch; locally isometric embedding; anisotropic remeshing; bijective parameterizations
CLC number: TP391.41 **Document code**: A
2010 Mathematics Subject Classification: 65D11

1 Introduction

Anisotropic remeshing is an important problem in computer graphics[1], scientific computing[2], fluid simulation[3], etc. It has extensive and important applications in many fields, such as geometric processing, mechanical manufacturing, and physical simulation. Namely, anisotropic remeshing requires that the shapes, sizes, and orientations of mesh elements conform to the input Riemannian metrics in advance.

Compared to isotropic remeshing, anisotropic remeshing has more advantages. For example, in fields of interpolation error control and finite element analysis[2,4], the anisotropic remeshing requires fewer mesh elements than the isotropic remeshing to achieve the same level of accuracy.

Anisotropic remeshing should satisfy the following requirements:

(Ⅰ) The sizes and orientations of mesh elements should conform to the given Riemannian metric, that is, the image of each element under the inverse transformat.

(Ⅱ) The minimum angles of the images of mesh elements under the inverse transformation is as large as possible; otherwise, numerical problems will occur in subsequent applications (such as finite element analysis).

(Ⅲ) The approximation error of the anisotropic mesh to the input mesh should be as small as possible. Furthermore, it should retain the features of input meshes (see Fig. 1).

The above requirements involve many complex and coupled nonlinear constraints. Thus, it is very challenging to generate an anisotropic mesh that satisfies all of the requirements. One possible idea is based on

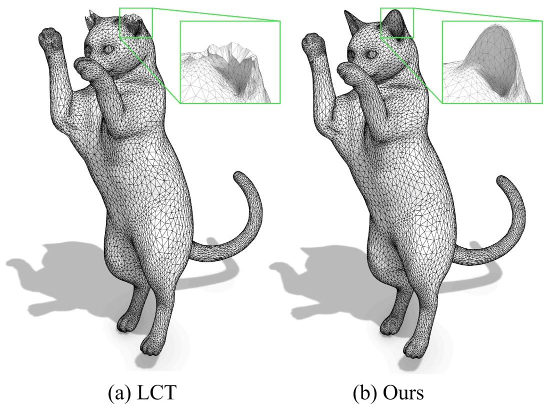


Fig. 1 (a) The result of the LCT algorithm[5]. (b) our result. It can be seen that the LCT algorithm misses the geometric features at the ear of the model, and the approximation error is large. Our result retains the features, and approximates to the input mesh better.

the embedding[6-10]. First, it embeds the input mesh into a target space where performing the remeshing is relatively easier, and then performes the remeshing in the target space, and finally transforms the mesh back to the original space using the inverse map of embedding. However, existing embedding algorithms have unavoidable defects. Zhong et al.[6] conformally embeds the mesh into a parametric 2D domain. But this embedding method could not guarantee bijection, so that it is difficult to get the inverse of embedding. Besides, it is complex to process the high-genus meshes, thereby affecting the robustness of the algorithm. In Ref. [7], the input mesh is embedded into a high-dimensional Euclidean space for anisotropic remeshing. According to the Nash embedding theorem[11], which states that every Riemannian manifold can be isometrically embedded into some high-dimensional Euclidean space, such embedding always exists. However, if the input metrics have sudden discontinuities, computing a practical embedding may be challenging.

Our algorithm is also based on the idea of embedding. Unlike previous works, our method adopts the idea of locally isometric embedding. The basis of this idea is that the input mesh can always be segmented into a set of single connected patches, each of which can be isometrically parameterized into the plane. As long as the parameterized patches are remeshed anisotropically conforming to the given Riemannian tensor fields, we map the remeshed patches back to the input mesh to achieve the resulting anisotropic meshes. The advantages and contributions of our method include:

(Ⅰ) Our algorithm only needs to deal with the patches homeomorphic to the disc, so that all the surfaces of arbitrary genus can be remeshed using a unified process. It guarantees the robustness of the algorithm.

(Ⅱ) Since each patch is embedded one-to-one into the plane with low distortion, it theoretically guaranteed that the embedding is bijective.

(Ⅲ) The generated anisotropic meshes approximate the input surfaces with small errors, and we retain sharp features of the input surface.

A large number of experiments have shown the robustness of our method (see Fig. 2(b)). Compared to state-of-the-art methods, our anisotropic meshes approximate the input mesh better (see Fig. 1) and retain the features of the input mesh (see Fig. 1 and Fig. 2(a)).

2 Related work

2.1 Anisotropic mesh generation

An isotropic mesh can be generated by inserting Steiner points in poor-quality elements and recalculating the

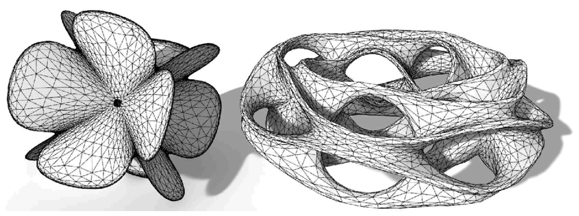


Fig. 2 There are many self-intersections on model (a). Our algorithm successfully generates ideal results. The model (b) is a high-genus mesh with multiple "holes", and our method also remeshes it correctly.

corresponding Delaunay triangulation. By modifying the criteria and methods of the point insertion, this method has also been successfully applied to anisotropic remeshing[2,12]. Another classic method for anisotropic remeshing is based on centroidal Voronoi diagram (anisotropic centroidal Voronoi tessellation, ACVT)[13-16]. This method uniformly samples points on the mesh to generate the corresponding anisotropic Voronoi diagram (AVD), and moves the points to the center of gravity of the Voronoi diagrams. It iterates the process until convergence. However, computing anisotropic Voronoi diagram is very expensive, thereby seriously affecting the efficiency and robustness.

The particle-based mesh generation methods[17-19] regard the mesh vertices as particles. They first define the repulsive or elastic forces between the particles, and then update the particles until the static equilibrium is achieved. According to the different energy definitions, the final mesh with different properties are generated. However, these methods strongly depend on the selection of parameters, and the results of different parameters may vary widely.

The quality of the anisotropic mesh can be measured by the difference between the solution u of the corresponding partial differential equation and the piecewise linear interpolation function $\widehat{u}$[20]. Previous work proposed the optimal Delaunay triangulation (ODT), which generates the anistropic mesh by minimizing the error function $\| u-\widehat{u} \|_{L^p(\Omega)}$[21]. This idea is also applied to anisotropic remeshing works. A representative one is the mesh generation algorithm using local convex functions[5]. It defines a convex function locally in each simplex and does not require u to be a global convex function. However, the algorithm includes a non-smooth projection operation in 3D space, which may cause a large approximation error between the resulting mesh and the input mesh.

Embedding the input mesh into another space for mesh generation[6-10] is another popular way. The input mesh is conformally embedded into the plane based on the uniformization theorem[6]. Then a weighted centroidal Voronoi tessellation (WAVT) and its Delaunay triangulation are computed on the plane. Finally, the 2D mesh is mapped back to the original space to generate the anisotropic mesh. However, it is not robust to remesh the high-genus models. An intersection-free high-dim embedding of the input mesh is computed[7] such that the pullback metric of the embedding matches the Riemannian metric. However, the solution of this embedding depends on a non-global optimization problem, which may fall into the local minimum. Besides, if the Riemannian metric is not smooth enough, computing such a non-intersecting embedding is challenging.

2.2 Parameterizations-based mesh generation

Computing parameterizations is a highly researched topic[22,23]. With the help of parameterization, the remesh can be carried out in the parameterized domain. The centroidal Voronoi diagram algorithm[24-26] on the 2D parameter domain is used to generate the mesh, but the calculation of Voronoi diagram is time-consuming, thus affecting the performance and efficiency of the solution. Similar to the locally embedding idea of our work. Ref. [26] adopts a local parameterization method to generate a high genus isotropic mesh. The remeshing method based on local parameterization has two difficulties: ① parameterizations distortion greatly affects the quality of remeshing; ② the remeshing quality near the patch boundary is poor. Our locally embedding algorithm resolves these two difficulties.

3 Method

3.1 Overview

Inputs Given a triangle mesh $M \in \mathbb{R}^3$, with $\mathscr{V}, \mathscr{T}, \mathscr{E}$ denoting the set of vertices, faces, edges, respectively, we define the Riemannian metric $\mathscr{M}$ on the input mesh. Specifically, the Riemannian metric of a point p is defined as a 3×3 symmetric positive definite (SPD) matrix $\mathscr{M}_p$. Let $U\Sigma U^{\mathrm{T}}$ indicate the Singular Value Decomposition of $\mathscr{M}_p$, where U is an orthogonal matrix and Σ is a diagonal matrix. The $\mathscr{Q}=U\Sigma^{\frac{1}{2}}U^{\mathrm{T}}$ implies the inverse transformation, which transforms anisotropic space to isotropic space. On the other hand, a target edge length L_{tar} is specified. After mapping the generated anisotropic mesh into the isotropic space, the edge length should be as close as possible to L_{tar}. Since the metric needs to be updated in the remeshing algorithm (Section 4.1), a reference mesh $M_{\mathrm{ref}} \in \mathbb{R}^3$ is also required. Usually we set $M_{\mathrm{ref}}=M$.

Outputs The output is a triangle mesh $\widehat{M}$, with the shape, size, and distribution of whose triangle face $\tilde{t} \in \tilde{\mathscr{T}}$ conforming to the metric $\mathscr{M}$ defined on the input mesh M. More precisely, the mapped triangle $\hat{t} = \mathscr{Q}\,\tilde{t}$ under the transformation $\mathscr{Q}$ should be a regular triangle.

Embedding methods The existing embedding algorithms, such as the globally conformal embedding[6], have to take the genus of the complex surfaces into consideration, and have no guarantee of the bijectivity. Thus, we propose an algorithm based on locally isometric embedding (see the Algorithm 1 and Fig. 3). Here, the locality means that the surface will be segmented into a set of simply connected patches and each patch will be remeshed subsequently. The isometry requires that length of any curve on the surface remains the same as much as possible before and after embedding in the Euclidian space.

Algorithm 1 Anisotropic mesh generation algorithm based on locally isometric embedding

Input: Triangle mesh $M \in \mathbb{R}^3$, Riemannian metric field $\mathscr{M}$

Output: Triangle mesh $\widehat{M} \in \mathbb{R}^3$, anisotropy is consistent with $\mathscr{M}$

```
1: while There are still unprocessed triangles in $M$ $t \in \mathscr{T}$ do
2:     // Cone patch generation (Section 3.2)
3:     a. Create a cone patch $P$ with $t$ as the center
4:     // Anisotropic Mesh Generation (Section 3.3)
5:     b. Parameterize $P$ to the 2D plane and get the mesh $\widehat{M}$
6:     c. Transfer metric $\mathscr{M}$ to 2D parameter domain
7:     d. Anisotropic mesh generation, making $\widehat{M}$ consistent with metric $\mathscr{M}$
8:     e. Map $\widehat{M}$ back to the original space using the inverse map, replace $P$, and update the mesh $M$
9: end while
10: // minimum angle increase (Section 3.4)
11: while There are triangles of poor quality $t$ in $M$ do
12:     Post-processing the surface patches near $t$ to improve the quality of the elements
13: end while
```

Challenges However, there are two major difficulties in our algorithm with locally isometric embedding:

(Ⅰ) Before the embedding, the input mesh needs to be segmented into a collection of simply connected patches. It is a challenging to make sure that each patch can be embedded into the target space with very low isometric distortion.

(Ⅱ) The patches after remeshing should be stitched together. The inter-compatibility of the connectivity between different patches is another issue to be addressed.

Key ideas

(Ⅰ) To embed each patch into the target space with low distortion, we require that the patch approximates the cone surface.

(Ⅱ) To optimize the boundary region of each patch effectively, we process the patches in sequence. The remeshed patch P_{prev} will be mapped back to the source surface M to update the corresponding part of the input mesh. Then we select the next patch P_{cur}, which has an

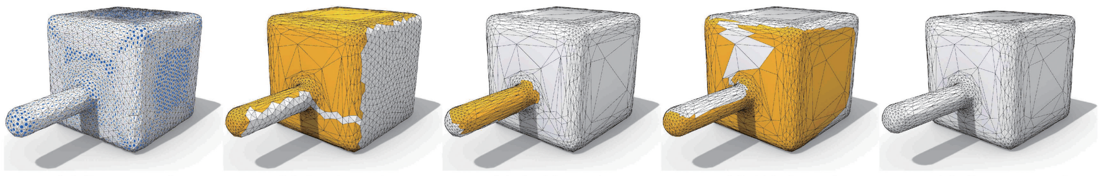


Fig. 3 Algorithm pipeline. The dark blue ellipse is the visualization of the Riemannian tensor field; the yellow part is the currently processed patch. The figure shows the results after the first, third, and seventh iterations, respectively. The remeshing process is done piece by piece, and there are intersections between the current cone patch and the processed area, which ensures that the elements near the boundary could be remeshed correctly.

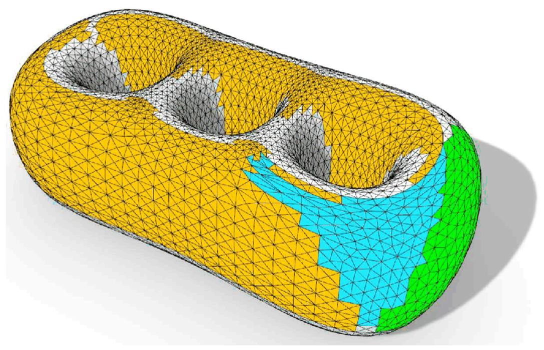


Fig. 4 Boundary treatment. $P_{\mathrm{prev}}=\Omega_{\mathrm{green}}\cup\Omega_{\mathrm{blue}}$, $P_{\mathrm{cur}}=\Omega_{\mathrm{yellow}}\cup\Omega_{\mathrm{blue}}$, There is a common part between the two surfaces $\Omega_{\mathrm{blue}}=P_{\mathrm{prev}}\cap P_{\mathrm{cur}}$, The unprocessed triangle patches in Ω_{blue} will be processed in P_{cur}.

intersection with the interior of P_{prev}. This ensures that the unprocessed boundary region of P_{prev} in the previous step is optimized in current step (see Fig. 4). Here, P_{prev}, P_{cur} represents the processed, and to be processed patch, respectively.

3.2 Cone surface generation

Cone surface is a kind of developable surface[27-29], whose Gaussian curvature is zero everywhere. The cone surface has two basic components: the central axis and the angle between the surface normal and the axis, which are denoted by $\boldsymbol{n}_P$, α_P, respectively. We define the following cost term:

$$E(P,\ t)=\ (\langle \boldsymbol{n}_P,\ \boldsymbol{n}_t\rangle-\cos\alpha_P)^2 \qquad (1)$$

For a given surface P and any triangle t with unit normal n_t, the cost term $E(P,t)$ measures compatibility between them. If the cost is small, the triangle is consistent with the developability of the surface P.

The generation process of a cone patch is as follows: First, $\boldsymbol{n}_P$, α_P are needed for generating a patch. A vertex is randomly selected on the input mesh, and the 1-ring faces of the vertex Ω_f are used to calculate $\boldsymbol{n}_P$, α_P. Specifically, the corresponding $\boldsymbol{n}_P$ and α_P are obtained by solving an optimization problem with nonlinear constraints as follows:

$$\min_{n_P,\alpha_P}\frac{1}{A_{\Omega_f}}\sum_{t\in\Omega_f}A_t\,E(\Omega_f,t),\ \text{s.t.}\ \ \|\boldsymbol{n}_P\|^2=1 \qquad (2)$$

where A_{Ω_f} denotes the patch area. Then traverse every triangle face t in Ω_f and finally P is initialized with the triangle with the smallest cost $E(\Omega_f,t)$. We also call the triangle t the seed of the cone patch.

To expand P, new triangles need to be added to P. Thus, the following metrics are defined:

$$C(P,\ t)=\frac{E(P,t)\ D(s_P,\ t)^{2\beta}}{A_P} \qquad (3)$$

where s_P represents the seed triangle of P, $D(s_P,\ t)$ represents the shortest path inside the patch between the two triangles, and β is set 0.7 by default.

P is expanded according to the following rule: Insert the adjacent triangles of the seed into a priority queue Q. Q is sorted by $C(P,t)$ in ascent order. As long as Q is not empty, the top triangle with minimal cost can be popped. If its cost is less than a threshold $C_{\max}$, add it to the cone patch and push its adjacent triangles into Q. Otherwise, the patch expansion finishes. The threshold $C_{\max}$ controls the approximating quality to the developability. It is set 0.1 by default. Note: It is critical to prevent generating the loop structure when adding triangles so that the final patch is simply connected. Specifically, we store the edges and vertices of P in set $\mathscr{E}_P$, $\mathscr{V}_P$. Before element t is added in, we check whether all the three vertices and only one edge of t are already in $\mathscr{V}_P$, $\mathscr{E}_P$, if not, there is no loop structure and t can be added into P.

3.3 Planar anisotropy mesh generation

Embedded algorithm Theoretically, a cone surface can be embedded in a plane without distortion. In this paper, the bijective parameterization method based on the obstacle function of the triangle inequality[30] is used.

Riemannian metric transfer To generate the anisotropic mesh in 2D space, the Riemannian metric defined on the vertices of the 3D mesh is transfered to the 2D parameter domain with as low distortion as possible. This is ensured by our algorithm: the simply connected surface we generated is highly developable so the result has very low distortion, which makes the transfer of the metric from 3D to 2D with almost no distortion.

For a vertex $\mathrm{v}=(v_x, v_y, v_z)$ of the 3D mesh M, the corresponding metric and the parameter coordinate are defined as $\mathscr{M}$ and $\hat{v}=(\hat{v}_x, \hat{v}_y, 0)$. We solve the metric $\hat{\mathscr{M}}$ on $\hat{v}$. The metric $\mathscr{M}$ is decided by three principal directions d_1, d_2, d_3 and three eigenvalues k_1, k_2, k_3. Since the z-coordinate in the 2D parameter domain is 0, there are only two principal directions $\hat{d}_1$, $\hat{d}_2$ of $\hat{\mathscr{M}}$ that need to be solved. Consider unit orthogonal frames $O=(d_1, d_2, d_1\times d_2)$ and $\hat{Q}=(\hat{d}_1, \hat{d}_2, \hat{d}_1\times\hat{d}_2)$ on vertices v and $\hat{v}$. For any vertex v_i in the 1-ring of v, we define the vector $vv_i \overset{\triangle}{=} v_i-v$. Then, project it to the plane of d_1, d_2 and find the unit coordinates of the projected vector under the frame O, also recorded as vv_i. Repeat this step for 1-ring of $\hat{v}$ to get the unit vector $\widehat{vv}_i$. Finally traverse all the vectors in 1-ring Ω_v, and minimize the following error function:

$$\sum_{i\in\Omega_v} \| vv_i - \widehat{vv}_i \|^2 \tag{4}$$

using the least squares method to solve for the principal directions $\hat{d}_1$, $\hat{d}_2$. Let $D=(\hat{d}_1, \hat{d}_2, \hat{d}_1\times\hat{d}_2)$ and $K=\mathrm{diag}(k_1, k_2, k_3)$, then the metric $\hat{\mathscr{M}}=DKD^{\mathrm{T}}$.

Anisotropic mesh generation using local convex functions For the parametrized mesh $\hat{M}$, we use the anisotropic mesh generation algorithm[5] based on local convex functions for remeshing. The algorithm has three steps: updating the connectivity, optimizing the vertex positions and adjusting the edge lengths. The results can be seen in Fig. 5(c).

Mapping back to the source The positions and the connectivity on the boundary remain unchanged, so we just record the positions of vertices on the boundary before parameterization, and map them back directly. For the internal vertex v_{in}, we project it onto the initial 2D mesh before optimization. Without loss of generality, let v_{in} be projected on a triangle of t_{in}, and calculate its coordinate of the center of gravity $b_{\mathrm{in}}=(b_0, b_1, b_2)$, then the positions of v_{in} in the original space is

$$p_{\mathrm{in}} = b_0p_{0,\mathrm{in}} + b_1p_{1,\mathrm{in}} + b_2p_{2,\mathrm{in}},$$

where $p_{0,\mathrm{in}}$, $p_{1,\mathrm{in}}$, $p_{2,\mathrm{in}}$ are the coordinates of the vertices of t_{in} in 3D space.

3.4 Minimum angle improvement

The anisotropic meshes generated by the above steps may have triangles with large deviations from the given Riemannian metric, such that the triangles often have small angles, which can greatly affect the stability of the solver of finite element method and error analysis. In this paper, the quality of these triangles is improved by optimizing the vertex positions and connectivity. See Algorithm 2 for the procedure.

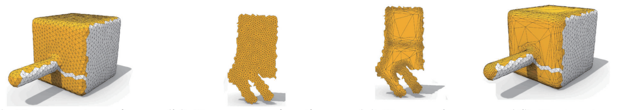

(a) Cone extraction (b) Parameterization (c) Remesh (d) Output

Fig. 5 One iteration process. The figure shows the process of selecting a cone surface patch and remeshing it. The yellow part represents the currently processed patch.

Algorithm 2 Minimum angle improvement

```
Input: Triangle mesh $M\in\mathbb{R}^3$, Riemannian metric field $\mathscr{M}$
Output: Triangle mesh $M^*\in\mathbb{R}^3$, which has the larger minimum angle
1: while $q(t)$ of $t\in\mathscr{T}$ is below a threshold and the number of iterations does not reach the upper bound do
2:     a. Generate a cone surface patch $P$ over a number of neighbors centered on $t$
3:     b. Parameterize $P$ to the 2D domain and get the mesh $\hat{M}$
4:     c. Transfer the metric $\mathscr{M}$ to the 2D parameter domain
5:     d. Update the connectivity and optimize the vertex positions on $\hat{M}$, see later
6:     e. Map $\hat{M}$ into the original space using parametric inversion, and update $P$ and $M$
7: end while
```

Quality evaluation For any triangle facet $t\in\mathscr{T}$, it is mapped into a facet $\tilde{t}$ in isotropic space using the inverse transformation induced by the Riemannian metric. Define the quality

$$q(t) = \tilde{q}(\tilde{t}) \overset{\triangle}{=} \frac{2\sqrt{3}a}{ph} \in (0, 1],$$

where a, p and h are the area of $\tilde{t}$, half the perimeter of $\tilde{t}$, the longest length of $\tilde{t}$, respectively. The closer $\tilde{t}$ is to an equilateral triangle, the greater $q(t)$. And $q(t)=1$ if and only if $\tilde{t}$ is an equilateral triangle. θ indicates the angles of $\tilde{t}$, and obviously, the closer θ is to 60°, the higher the quality of t.

Vertex position update For $t\in\mathscr{T}$, $\mathscr{M}_1$, $\mathscr{M}_2$, $\mathscr{M}_3$ are the Riemannian metrics of its vertices, then the metric of t is $\mathscr{M}_t=(\mathscr{M}_1+\mathscr{M}_2+\mathscr{M}_3)/3$. Let the SVD of $\mathscr{M}_t$ is $\mathscr{M}_t = U\Sigma V^{\mathrm{T}}$, $\Lambda = \Sigma^{\frac{1}{2}}$, the inverse transformation $\mathscr{Q}_t=U\Lambda V^{\mathrm{T}}$ induced by $\mathscr{M}_t$, which can map t in anisotropic space to $\tilde{t}$ in isotropic space. Ideally, $\tilde{t}$ should be an equilateral triangle. Let p denote the current optimized vertex, we define the following energy:

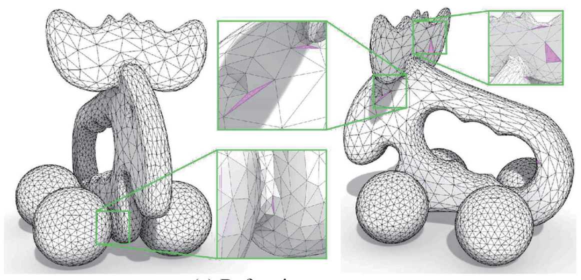
$$Q_t = \mathrm{e}^{w(\|J\|_F^2+\|J^{-1}\|_F^2)} \tag{5}$$

$$Q_p = \sum_{t\in\Omega_p} Q_t \tag{6}$$

where J is the Jacobi matrix between $\tilde{t}$ and a standard equilateral triangle with edge length L_{tar}. Using the gradient descent $p_{\text{new}}=p-\alpha\nabla_p Q_p$, if the energy descends and the triangle is not flipped, we update the position of p.

Connectivity update The topological operation we apply in this process only contains edge-flipping. The above energy Q_t can be used to determine whether the edge is flipped, and the principle of maximizing the minimum angle can be used as well. Repeat the algorithm until the mass Q_t of all triangles or the number of iterations reaches the specified values. The number of small angles will be reduced by the minimum angle improvement (see Fig. 6).

Discuss The idea of minimal angle improvement is proposed in an isotropic remeshing method: EBFR method[31]. As mentioned in many papers, minimal angle improvement is essential for many simulation applications[2,4,31]. Although there are only a few small angles, it deteriorates the stability of the solver in FEM and the error bound in the interpolation problem. However, EBFR method fails for some models because it may run into infinite loop when greedily improving the minimal angle[32].


(a) Before improvement

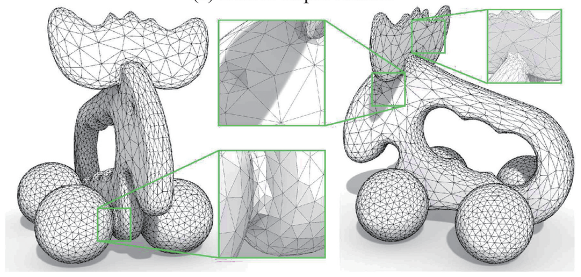

(b) After improvement

Fig. 6 The minimum angle improvement. Magenta represents the triangle faces with the minimum angle less than 20°. (a) There are 10 magenta triangle faces without the minimum angle improvement. (b) After the improvement, the number of such faces is 0.

4 Experiments

Our algorithm is implemented based on C++, and the experiments are all completed in a desktop with Windows 10 operating system, 3.20 GHz Intel Core i7-8700 CPU, 16 GB RAM. A large number of experiments are carried out in this paper, and we discuss and compare our method with LCT algorithm[5], conformal embedding algorithm[6], partical-based algorithm[19], L_p CVT algorithm[15] and obtuse angles removing alrotithm[34].

4.1 Metric computation and evaluation

Riemannian metric generation The Riemannian metric on each vertex during the remeshing process is obtained by linearly interpolating the Riemannian metric of the reference mesh M_{ref}. Assuming the current vertex $v\in\mathscr{V}$, projecting v onto the nearest facet t on M_{ref}, the corresponding barycentric coordinates are b_0, b_1, b_2. Then, we denote the Riemannian metrics of vertices of t as $\mathscr{M}_0$, $\mathscr{M}_1$, $\mathscr{M}_2$, and the Riemannian metric of v, $\mathscr{M}_v$, can be formulated by barycenter interpolation: $\mathscr{M}_v=b_0\mathscr{M}_0+b_1\mathscr{M}_1+b_2\mathscr{M}_2$.

The Riemannian metric of M_{ref} is obtained by curvature. We denote k_1 and k_2 as principal curvatures and d_1, d_2 as corresponding principle directions, then the corresponding Riemannian metric

$$\mathscr{M}=(d_1, d_2, d_1\times d_2)\operatorname{diag}(\max(|k_1|, 10^{-4}), \max(|k_2|, 10^{-4}), 0)\,(d_1, d_2, d_1\times d_2)^{\mathrm{T}},$$

where 10^{-4} is used to prevent the long edge length caused by too small curvature.

Quality evaluation criterion In addition, in order to evaluate the quality of the mesh, the relevant indicators need to be quantified. For facet $t\in\mathscr{T}$ and the corresponding facet $\tilde{t}$ in the isotropic space, we use the following metrics to evaluate the quality of t:

① The quality of facet

$$q(t)=\tilde{q}(\tilde{t})\overset{\triangle}{=}\frac{2\sqrt{3}a}{ph}\in(0, 1],$$

where a is the area of $\tilde{t}$, p is the semi-perimeter of $\tilde{t}$, and h is the length of the longest edge of $\tilde{t}$. The closer $\tilde{t}$ is to a regular triangle, the larger the value of $q(t)$, and $q(t)=1$ if and only if $\tilde{t}$ is a equilateral triangle.

② The minimum angle

$$\theta(t)\overset{\triangle}{=}\min(\deg_{\tilde{t}0}, \deg_{\tilde{t}1}, \deg_{\tilde{t}2}),$$

where $\{\deg_{\tilde{t}i}\}_{i=0}^{2}$ correspond to three angles of $\tilde{t}$. The closer $\theta(t)$ is to 60°, the better the quality of the triangular face t.

③ We uses all angles θ (not only the minimum angle) and frequency histograms hist(θ), hist(q) for visual analysis.

As stated in Sec. 3.1, $q(t)$, $\theta(t)$ are both calculated via the inverse transformation induced by the input Riemannian metric, so that the better the value of quality $q(t)$, $\theta(t)$, the smaller the difference between output and the input metric. In fact, our post-processing is enlightend by AMIPS method[33], which can effectively penalize the maximal distortion between the source and the target.

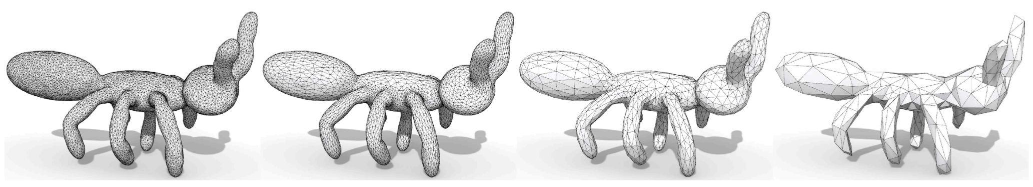


Fig. 7 Results of different target edge lengths. (a) Input mesh. The target edge length of (b), (c), (d) is 1.5 times, 3.0 times, 6.0 times L_{avg}, respectively. Here, L_{avg} represents the average edge length of the mesh (image under inverse transformation) and L_{tar} indicates the target edge length.

4.2 Algorithm analysis

Target edge length An advantage of anisotropic meshes compared to isotropic meshes is that they can use fewer numbers of faces to achieve the same level of accuracy. We get different results for different target edge lengths (See Fig. 7). The leftmost side is the input mesh, and the target edge length L_{tar} of the results on the right hand side is 1.5, 3 and 6 times of the average edge length L_{avg}, respectively. It can be seen in the figure that the result of 1.5 times is almost the same as the input visually, some details (such as tentacles) in the result of 3 times blur, and some details (such as the torso) of 6 times are lost.

Quality of input mesh We use several input meshes of different qualities for testing, while these meshes represent the same model. As shown in Fig. 8, the first row contains inputs with four different qualities, and the second row are the corresponding outputs. It can be seen that whether the input is isotropic of high quality or with high noise, we always get acceptable results. It demonstrates our algorithm is robust. In this experiment, the target edge length is set to 0.1.

Expansion of cone patches In order to ensure that there is a common intersection between different cone patches, our method selects a center point on the boundary of the surface patch that has been processed, and generates a new cone patch with this center point. The advantage of this procedure is that there it does not need to explicitly detect the collision of the boundaries. However, sometimes it can not guarantee well-processed triangle facets in the boundary. If the region near the center point has poor developability and high distortion, the generation of cone surface patches may be obstructed, which may affect the coverage between patches. For this reason, after the cone patch generation, we exapnd 1 − 2 neighborhoods in the boundary which increases the area of the intersection region with the processed surface, so that the boundary area can be processed better. Fig. 9 shows the effect of subsequent expansion on the results. Fig. 9 (a) does not perform subsequent expansion, so it can be seen that in some areas with poor developability, the mesh quality is poor. The red triangles in the green box are near degeneration, and the minimum of the metrics θ, q are less than 0.05, 2°, respectively, which is not conducive

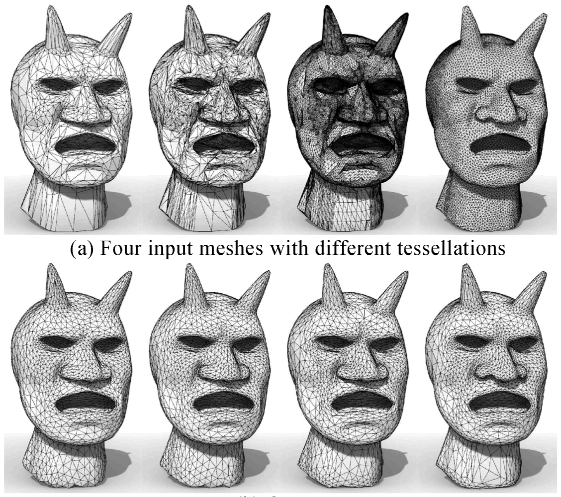


Fig. 8 The results of the same surface with different tessellations.

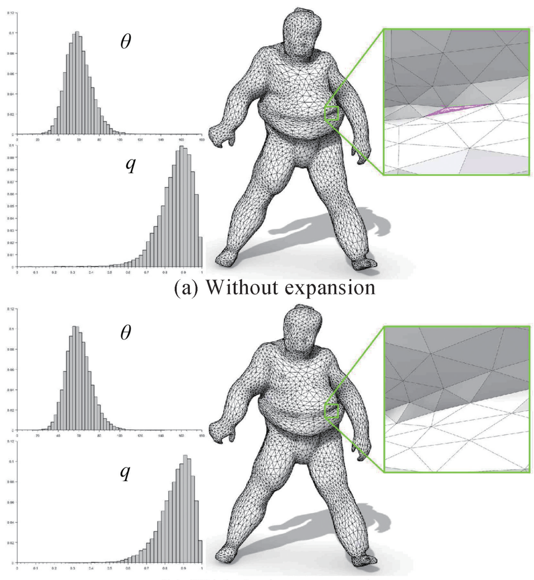
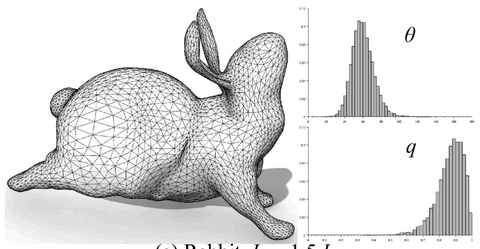
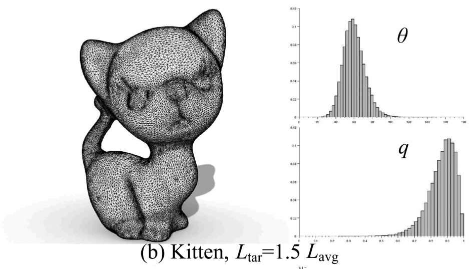
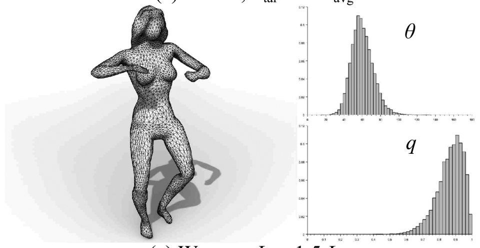
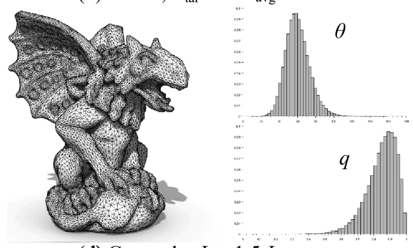


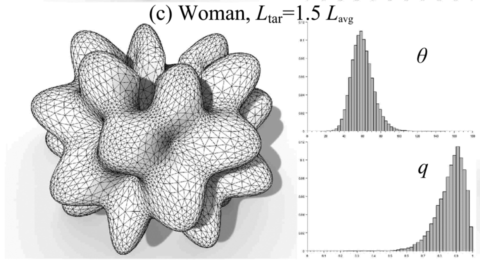
Fig. 9 The effect of whether the subsequent expansion of the surface patch is performed. (a) shows the result of no expansion, and the triangle facets in some region (see the green frame) is close to degeneration. This is because of the bad developability of the patch nearby, so that the boundary part is not fully processed. (b) is the result of the expansion of 1-ring. It can be seen that the pacth has been fully processed, and the quality of the triangular faces reaches a high level.


(a) Rabbit, L_{tar}=1.5 L_{avg}


(b) Kitten, L_{tar}=1.5 L_{avg}


(c) Woman, L_{tar}=1.5 L_{avg}


(d) Gargoyle, L_{tar}=1.5 L_{avg}


(e) Torus, L_{tar}=1.5 L_{avg}

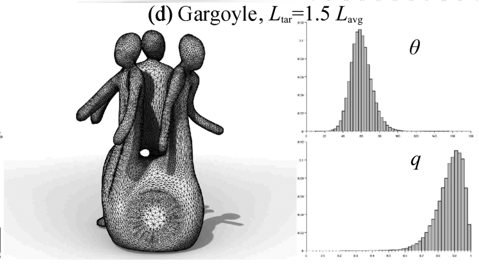
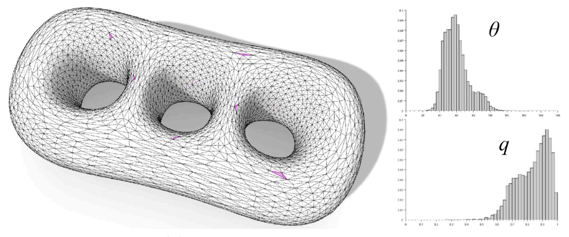

(f) Memento, L_{tar}=1.5 L_{avg}

Fig. 10　More results.


(a) LCT

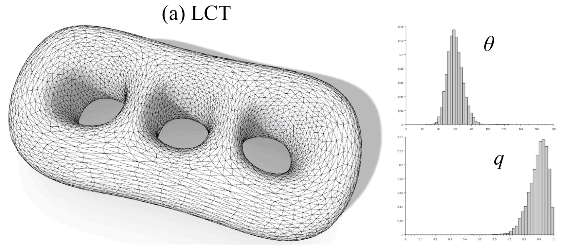
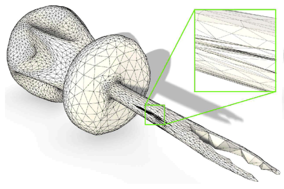

(b) Ours

Fig. 11　(a) The result of the LCT algorithm. (b) Our result. Neither performs the minimum angle improvement. The minimum angle of the red triangular facet is θ<25°. The right hand side is the histogram of all angles θ and quality q. It can be seen that θ is more concentrated around 60°, and the quality q is all above 0.4 and more close to 1 in our algorithm.


(a) LCT

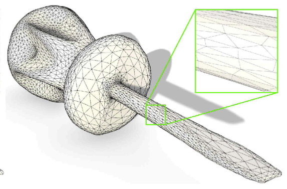

(b) Ours

Fig. 12　(a) The result of the LCT algorithm. (b) Our result. In this comparison, neither performs the minimum angle improvement. The LCT algorithm fails to retain the sharp features of the input model, which further deteriorates the quality of triangular facets in the nearby area (see the area in the green frame). Our method not only has smaller approximation errors to the input model, but also maintain the quality of triangular facets at a high level.

to the subsequent processing and application of the model. Fig. 9 (b) is the result of 1-ring expansion, where the quality of triangle facet is higher, and the minimum of θ, q is not less than 0. 25, 15°, respectively. But this does not mean that the larger the cone patch, the better, because a lager patch may have a poor developability. Experiments verify that it is generally sufficient to expand about 1−2 neighborhoods of the patch.

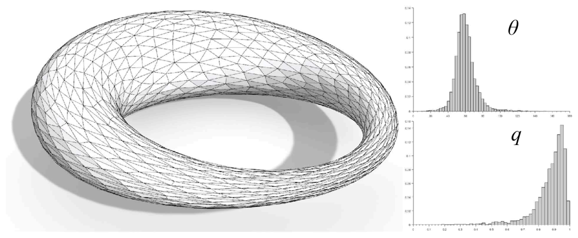
More models　As shown in Fig. 10, the results of our method are highly consistent with the Riemannian metric. The metric θs are concentrated about 60°, and qs are mostly above 0. 2, concentrated about 1. Statistics of a series of remeshing quality metrics and the timings for all the demonstrated examples are reported in Table 1. Fig. 11 shows a comparison between our method and the LCT method[5]. For a fair comparison, we use the same input mesh, and the target edge lengths are both set to 0.692757 (1. 5L_{avg}). Our method generates triangle factes which are more uniformly distributed in anisotropic regions, and the shapes and sizes of which in different areas vary more smoothly and naturally. As shown in the histogram, the results of our method are more consistent with the given Riemannian metric.

4.3 Comparison

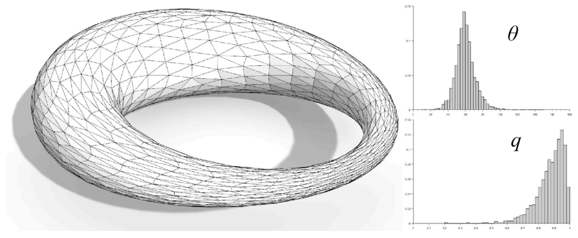
Fig. 12 shows another comparison between our method and the LCT algorithm. For a fair comparison, we use the same input mesh and the target edge lengths are both set to 0.0832209 ($1.5L_{avg}$). The mesh shape is a screwdriver, whose front is thin and sharp. The approximation error of the LCT result to the input mesh is very large in these areas. For example, a "gap" appears in the front of the model, but the result of our method can approximate the input mesh well.

Tab. 1 Statistics. The table records the number of vertices and faces of the input/output mesh, the target edge length (L_{tar}), other related measures and the running time in seconds for all the models. q_{min}, q_{max}, q_{avg} denotes the minimum, maximum, and average of the triangular quality measure, respectively. θ_{min}, and θ_{max} denote the minimum, maximum angle of the triangle after the inverse transformation, respectively. θ_{avg} represents the average of the minimum angles among all the triangles. Note that the experiments don't perform minimum angle improvement by default, except for those specified separately.

Model	$\#V_{input}$	$\#F_{input}$	$\#V_{output}$	$\#F_{output}$	L_{tar}	$q_{min}/q_{max}/q_{avg}$	$\theta_{min}/\theta_{max}/\theta_{avg}$	time(s)
Cat Fig. 1	27894	55712	10841	21606	$1.5L_{avg}$	0.032/0.999/0.848	1.89°/175.65°/47.70°	56.246
Focal octa Fig. 2(a)	13000	25996	10926	21848	$1.1L_{avg}$	0.018/0.998/0.755	1.06°/175.43°/41.66°	36.831
Heptoroid Fig. 2(b)	49958	100000	5393	10870	$3.0L_{avg}$	0.113/0.997/0.825	7.08°/164.91°/46.03°	63.534
Fig. 3 (e)	2309	4614	1554	3104	$1.35L_{avg}$	0.130/0.998/0.828	7.94°/162.69°/46.25°	3.598
Holes3 Fig. 11(a)	5884	11776	4463	8934	$1.1L_{avg}$	0.261/0.999/0.835	17.01°/145.70°/48.04°	3.363
Holes3 Fig. 11(b)	5884	11776	4425	8858	$1.1L_{avg}$	0.422/0.998/0.893	26.64°/125.14°/50.97°	9.666
Elk Fig. 6(a)	5194	10388	4285	8570	$1.1L_{avg}$	0.157/0.998/0.864	9.92°/159.14°/48.82°	13.377
Elk Fig. 6(b)	5194	10388	4285	8570	$1.1L_{avg}$	0.383/0.998/0.864	20.84°/129.96°/48.79°	13.377
Ant Fig. 7(b)	13000	25996	5582	11160	$1.5L_{avg}$	0.254/0.998/0.873	10.45°/145.86°/49.46°	19.375
Ant Fig. 7(c)	13000	25996	1256	2508	$3.0L_{avg}$	0.184/0.998/0.834	9.10°/154.71°/46.76°	10.429
Ant Fig. 7(d)	13000	25996	302	600	$6.0L_{avg}$	0.174/0.997/0.757	6.63°/152.63°/41.36°	9.658
Oni1 Fig. 8-1	1435	2866	2390	5856	0.1	0.061/0.998/0.850	3.90°/171.87°/47.76°	6.564
Oni2 Fig. 8-2	4435	8866	3237	6470	0.1	0.215/0.998/0.848	13.27°/151.49°/47.69°	9.542
Oni3 Fig. 8-3	28930	57856	3032	6060	0.1	0.059/0.998/0.769	3.17°/171.63°/42.26°	29.376
Oni4 Fig. 8-4	13000	25996	3262	6520	0.1	0.059/0.997/0.842	3.34°/172.05°/47.17°	12.967
Sumoroti Fig. 9(a)	13000	25996	5714	11424	$1.5L_{avg}$	0.037/0.999/0.853	1.91°/175.03°/48.03°	20.224
Sumoroti Fig. 9(b)	13000	25996	5619	11234	$1.5L_{avg}$	0.275/0.999/0.858	15.79°/143.00°/48.35°	17.836
Screwdriver Fig. 12(a)	13000	25996	7834	15664	$1.5L_{avg}$	0.015/0.999/0.863	0.88°/177.92°/48.88°	9.496
Screwdriver Fig. 12(b)	13000	25996	4185	8366	$1.5L_{avg}$	0.254/0.998/0.860	13.07°/146.57°/48.45°	13.685
Rabbit Fig. 10(a)	13000	25996	5656	11308	$1.5L_{avg}$	0.359/0.998/0.865	19.55°/133.04°/48.84°	17.487
Kitten Fig. 10(b)	50000	100000	20350	40700	$1.5L_{avg}$	0.253/0.999/0.865	15.65°/146.60°/48.85°	76.478
Woman Fig. 10(c)	13000	25996	5149	10294	$1.5L_{avg}$	0.346/0.998/0.864	17.39°/134.13°/48.86°	16.171
Torus Fig. 10(e)	16815	33630	7578	15156	$1.5L_{avg}$	0.236/0.999/0.865	15.32°/148.85°/48.92°	26.774
Memento Fig. 10(f)	49968	99932	19319	38634	$1.5L_{avg}$	0.203/0.999/0.869	10.45°/153.21°/50.46°	85.176
Cyclide Fig. 13(a)	21600	43200	1000	2000	–	0.192/0.997/0.877	9.38°/151.12°/50.24°	63.15
Cyclide Fig. 13(b)	21600	43200	928	1856	$3.0L_{avg}$	0.215/0.998/0.878	11.90°/150.57°/49.97°	16.453
Fertility Fig. 14(a)	13971	27954	12095	24202	$1.0L_{avg}$	0.347/0.999/0.881	20.26°/134.65°/50.07°	25.178
Gargoyle Fig. 10(d)	50002	100000	20467	40930	$1.5L_{avg}$	0.169/0.999/0.847	9.15°/156.41°/47.52°	128.538
Gargoyle(LCT)	50002	100000	18865	37726	$1.5L_{avg}$	0.008/0.999/0.838	0.52°/178.90°/47.13°	70.303

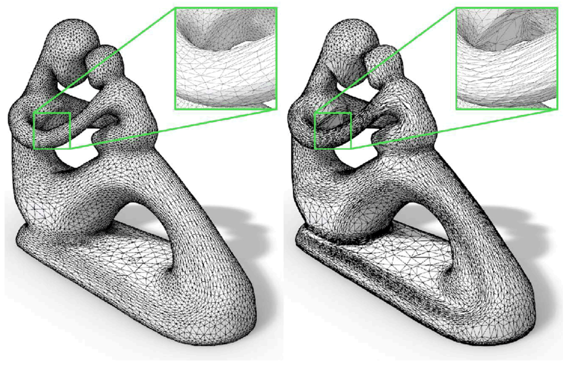


(a) Conformal Embedding



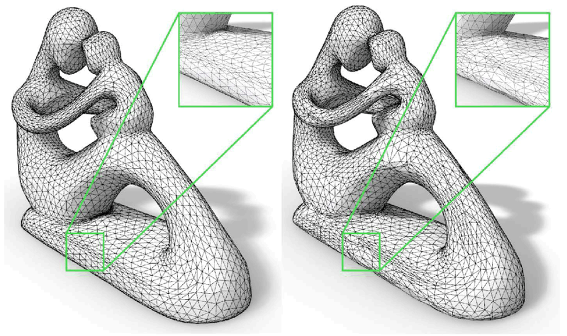
(b) Ours

Fig. 13 (a) is the result of conformal embedding method, (b) is the result of ours. The two methods are comparable, however, our method is more concise and time-saving since CVT in conformal embedding method is more time-consuming.



(a) Ours　　(b) Particle-based method

Fig. 14 (a) is the result of ours, (b) is the result of particle-based method.



(a) Input　　(b) Obtuse angles removed

Fig. 15 (a) is the input model, (b) is the result of removing obtuse angles.

Fig. 13 shows the comparison of our method and conformal embedding method[6]. Our result is comparable with their method. However, our method is more concise since we need only deal with disk-topology patch. And our method is more time-saving than theirs since CVT in conformal embedding method is more time-consuming. Since we don't have the source code of conformal embedding method, we take the time data in their paper for reference. As shown in Tab. 1, the total time of our result is even less than that of conformal parameterization (24 s in paper), to say nothing of the CVT time (39.15 s in paper).

Similarly, L_p Centroidal Voronoi Tessellation method (L_p CVT)[15] is also a CVT based method. Centroidal Voronoi tessellation (CVT) is a technique that has been applied to isotropic/anisotropic remeshing. However, L_p CVT method needs to calcaulate Voronoi diagram iteratively during the optimization, which is expensive and time-consuming, especially in anisotropic remeshing.

Fig. 14 compares our method and particle-based method[19], which applies a 6D metric in terms of vertex normals. The results indicate that the Riemannian metric of our mesh is closer to the input metric, which can be seen clearly from the figure. Our result yeilds more regular elements and the anisotropy coincides with curvature direction well.

Generally speaking, the closer an mesh element to an equilateral triangle under the inverse transformation, the better the quality. Hence some works investigate how to suppress large and small angles on isotropic meshes and further on anisotropic meshes. Xu et al.[34] propose a method to remove all the obtuse angles, which can be used as a post-processing step for anisotropic meshes generated from existing algorithms. Inspired by the mesh structrue from hexagon Minkowski metric, they propose to detect problematic metric hexagons called p-Hex, and based on which, they locally adjust the connectivity of the mesh while avoiding expensive Lloyd-type iterations. Although their method removes all the obtuse angles, the anisotropy may be destroyed. As shown in Fig. 15, the anisotropy of their result is not consistent with that of input, which is not suitable for applications which have higher requirements for the anisotropy quality. On the contrary, our angle improvement post-processing is premised on not destroying anisotropy, which maintains the anisotropy well.

5 Conclusions

We proposes an anisotropic mesh generation algorithm based on locally isometric embedding. Our algorithm is robust to remesh high-genus meshes and generates small approximation errors to the input mesh. As a consequence, our method retains the key geometric features of the input mesh. Moreover, our method can be regarded as a framework, in which we can try other geometry processing algorithms, but not limited to LCT method or even remeshing.

Our work also has some limitations. First, the process of selecting cone pathes is random. The results are different when the seed points are different. The second is that our method is not dominant in time. Since our method is serial, we have to deal with each patch using the same process, where we use LCT remeshing method, so that on average, our method is more costly than LCT method. We list time data for every model we used, as shown in Tab. 1, in which our method uses more time than LCT mothod on average, especially when the model is complex (model "Gargoyle" in Tab. 1), while more time-saving than CVT based method, such as conformal embedding method. Intuitively, the more patches and the more complex of model, the more time we consume. In order to overcome this limitation, it is worthwhile to study how to use parallel techniques to reduce the running time. In the future, we will generalize our geometry processing framework so that it can be used to solve more problems in computer graphics.

Acknowledgements

This work is supported by the USTC Research Funds of the Double First-Class Initiative(YD0010002003).

Conflict of interest

The authors declare no conflict of interest.

Author information

LI Huicong is currently a graduate student under the tutelage of assistant researcher Fu Xiaoming. His research interests focus on computer geometry, computer graphics and digital geometry processing.

FU xiaoming (corresponding author) is an assistant researcher in Graphics & Geometric Computing Laboratory, School of Mathematical Sciences, University of Science and Technology of China (USTC). He received his B. S. and Ph. D. degrees both from University of Science and Technology of China in 2011 and 2016, respectively. His research interests include geometric processing and optimization, CAD/CAE/IGA/Fabrication, VR/AR/MR and computer-aided geometric design. His research work can be found at his research website: http://staff. ustc. edu. cn/~fuxm/.

References

[1] Sun F, Choi Y, Wang W, et al. Obtuse triangle suppression in anisotropic meshes. Computer Aided Geometric Design, 2011, 28 (9): 537-548.

[2] Frey P J, George P L. Mesh Generation: Application to Finite Elements. Arlington, VA, USA: ISTE, 2007.

[3] Narain R, Samii A, O'Brien J F. Adaptive anisotropic remeshing for clothsimulation. ACM Transactions on Graphics, 2012, 31 (6):152.

[4] Shewchuk J R. What is a good linear element? Interpolation, conditioning, and quality measures. In: 11st International Meshing Roundtable. Springer, 2002: 115-126.

[5] Fu X M, Liu Y, Snyder J, et al. Anisotropic simplicial meshing using local convex functions. ACM Transactions on Graphics, 2014, 33 (6): 182.

[6] Zhong Z, Shuai L, Jin M, et al. Anisotropic surface meshing with conformal embedding. Graphical Models, 2014, 76 (5):468-483.

[7] Zhong Z, Wang W, Lévy B, et al. Computing a high-dimensionaleuclidean embedding from an arbitrary smooth riemannian metric. ACM Transactions on Graphics, 2018,37 (4): 62.

[8] Boissonnat J, Wormser C, Yvinec M. Anisotropic diagrams:Labelle Shewchuk approach revisited. Theoretical Computer Science, 2008, 408 (2/3): 163-173.

[9] Canas G D, Gortler S J. Surface remeshing in arbitrary codimensions. The Visual Computer: International Journal of Computer Graphics, 2006, 22 (9): 885-895.

[10] Kovacs D, Myles A, Zorin D. Anisotropic quadrangulation. Computer Aided Geometric Design, 2011, 28 (8): 449-462.

[11] Nash J. C^1 isometric imbeddings. Annals of Mathematics, 1954, 60 (3): 383-396.

[12] Dobrzynski C, Frey P. Anisotropic delaunay mesh adaptation for unsteady simulations. In: Proceedings of the 17th International Meshing Roundtable, 2008: 177-194.

[13] Du Q, Wang D. Anisotropic centroidal voronoi tessellations and their applications. SIAM Journal on Scientific Computing, 2005, 26 (3): 737-761.

[14] Valette S, Chassery J, Prost R. Generic remeshing of 3D triangular meshes with metric-dependent discrete voronoi diagrams. IEEE Transactions on Visualization and Computer Graphics, 2008, 14 (2): 369-381.

[15] Lévy B, Liu Y. L_p centroidal voronoi tessellation and its applications. In: SIGGRAPH '10: ACM SIGGRAPH 2010 papers, 2010: 119.

[16] Lévy B, Bonneel N. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In: Proceedings of the 21st International Meshing Roundtable, 2013: 349-366.

[17] Shimada K, Yamada A, Itoh T. Anisotropic triangulation of parametric surfaces via close packing of ellipsoids. International Journal of Computational Geometry and Applications, 2000, 10 (4): 417-440.

[18] Persson P, Strang G. A simple mesh generator in MATLAB. SIAM Review, 2004, 46 (2): 329-345.

[19] Zhong Z, Guo X, Wang W, et al. Particle-based anisotropic surface meshing. ACM Transactions on Graphics, 2013, 32 (4): 99.

[20] Zienkiewicz O, Taylor R, Zhu J. The Finite Element Method Set. 6th ed. Oxford, UK: Butterworth-Heinemann, 2005.

[21] Chen L, Sun P, Xu J. Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm. Mathematics of Computation, 2007, 76 (257): 179-204.

[22] Fu X, Liu Y, Guo B. Computing locally injective mappings by advanced MIPS. ACM Transactions on Graphics, 2015, 34 (4): 71.

[23] Zhang E, Mischaikow K, Turk G. Feature-based surface parameterization and texture mapping. ACM Transactions

on Graphics, 2005, 24 (1): 1-27.

[24] Alliez P, de Verdière E C, Devillers O, et al. Centroidal voronoi diagrams for isotropic surface remeshing. Graphical Models, 2005, 67 (3): 204 - 231.

[25] Alliez P, Meyer M, Desbrun M. Interactive geometry remeshing. ACM Transactions on Graphics, 2002, 21 (3). https://doi.org/10.1145/566654.566588.

[26] Surazhsky V, Alliez P, Gotsman C. Isotropic remeshing of surfaces: A local parameterization approach. In: Proceedings of 12nd International Meshing Roundtable. Springer, 2003: 215-224.

[27] Leopoldseder S, Pottmann H. Approximation of developable surfaces with cone spline surfaces. Computer-Aided Design, 1998, 30 (7): 571-582.

[28] Julius D, Kraevoy V, Sheffer A. D-charts: Quasi-developable mesh segmentation. Computer Graphics Forum, 2005, 24(3): 581-590.

[29] Pottmann H, Wallner J. Approximation algorithms for developable surfaces. Computer Aided Geometric Design, 1999, 16 (6): 539-556.

[30] Su J P, Ye C, Liu L, et al. Efficient bijective parameterizations. ACM Transactions on Graphics, 2020, 39 (4): 111.

[31] Hu K, Yan D, Bommes D, et al. Error-bounded and feature preserving surface remeshing with minimal angle improvement. IEEE Transactions on Visualization and Computer Graphics, 2017, 23 (12): 2560-2573.

[32] Cheng X X, Fu X M, Zhang C, et al. Practical error-bounded remeshing by adaptive refinement. Computers & Graphics, 2019, 82: 163-173.

[33] Fu X M, Liu Y, Guo B. Computing locally injective mappings by advanced MIPS. ACM Transactions on Graphics, 2015, 34 (4): 71.

[34] Xu Q C, Yan D M, Li W, et al. Anisotropic surface remeshing without obtuse angles. Computer Graphics Forum, 2019, 38 (7): 755-763.

基于局部等距嵌入的各向异性曲面网格生成

李慧聪,傅孝明*

中国科学技术大学数学科学学院,安徽合肥 230026

摘要: 提出了一种新颖的各向异性曲面网格生成方法. 不同于之前依赖于全局共形嵌入或高维等距嵌入的方法, 该算法以局部等距嵌入的思想为基础. 为了实现等距嵌入的目标,我们将原始曲面分割成圆锥曲面集,对曲面片逐一进行处理. 首先,利用双射参数化将圆锥曲面嵌入到平面,然后,在参数域进行各向异性网格生成,最后,将圆锥曲面映回原始曲面. 为了处理不同圆锥曲面之间的缝合问题,我们使当前圆锥曲面包含之前未处理的边界,使得边界附近的三角面片可以在当前迭代中处理. 大量实验验证了本文算法的鲁棒性. 相较于之前的各向异性网格生成算法,本文的算法能够更加鲁棒地处理高亏格网格,且能够得到与输入网格逼近误差更小的结果.

关键词: 黎曼度量; 圆锥曲面; 局部等距嵌入; 各向异性网格生成; 双射参数化