Quality Improvement of Segmented Hexahedral Meshes Using Geometric Flows

Juelin Leng, Guoliang Xu, Yongjie Zhang, and Jin Qian

Abstract This paper presents a new quality improvement algorithm for segmented quadrilateral/hexahedral meshes which are generated from multiple materials. The proposed algorithm combines mesh pillowing, curve and surface fairing driven by geometric flows, and optimization-based mesh regularization. The pillowing technique for quadrilateral/hexahedral meshes is utilized to eliminate doublets with two or more edges/faces located on boundary curves/surfaces. The non-manifold boundary for multiple materials is divided into several surface patches with common curves. Then curve vertices, surface vertices, and interior vertices are optimized via different strategies. Various geometric flows for surface smoothing are compared and discussed as well. Finally, the proposed algorithm is applied to three mesh datasets, the resulting quadrilateral meshes are well smoothed with volume and feature preserved, and hexahedral meshes have desirable Jacobians.

1 Introduction

In many applications such as computer graphics, finite element analysis and numerical simulations, 3D objects are usually discretized as polygonal meshes, typically tetrahedral and hexahedral meshes. For example, in biomedical modeling and material property analysis, 3D objects are often partitioned into multiple domains

J. Leng · Y. Zhang (✉) · J. Qian
Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jessicaz@andrew.cmu.edu

J. Qian
e-mail: jq@andrew.cmu.edu

J. Leng · G. Xu
ICMSEC, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China

J. Leng
e-mail: lengjl@lsec.cc.ac.cn

G. Xu
e-mail: xuguo@lsec.cc.ac.cn

Y.(J.) Zhang (ed.), *Image-Based Geometric Modeling and Mesh Generation*,
Lecture Notes in Computational Vision and Biomechanics 3,
DOI 10.1007/978-94-007-4255-0_11,

according to different physical/chemical attributes, or material properties. In computational simulations, quadrilateral and hexahedral meshes are often preferred [9]. For a segmented domain, the hexahedral mesh for each component composes the whole hexahedral mesh with conforming quadrilateral meshes on common surfaces. The union of boundary meshes for all components forms a non-manifold quadrilateral mesh. Compared to traditional mesh improvement problem for single-material domains, the problem is much more challenging for segmented meshes.

In this paper, we will focus on quality improvement of quadrilateral/hexahedral meshes for multiple materials. The pillowing technique for quadrilateral/hexahedral meshes is utilized to remove doublets. Then hexahedral mesh vertices are categorized into four types: fixed vertices, curve vertices, surface vertices, and interior vertices; while quadrilateral meshes only contain the first three types. Curve vertices and surface vertices are modified along the tangent directions to regularize the non-manifold boundary mesh. Moreover, geometric flows are applied for curve fairing and surface smoothing, which relocate curve and surface vertices along the normal directions. We will apply four typical geometric flows for surface smoothing, and discuss their effectivity of evolving the surface. Then the best feature-preserving geometric flow will be selected in our quality improvement algorithm. Interior vertices in hexahedral meshes are relocated via an optimization-based method. Finally, the proposed algorithms are validated on three application examples.

The rest of this paper is organized as follows. Section 2 reviews related previous work. We describe the quality improvement problem of quadrilateral/hexahedral meshes for multiple materials in Sect. 3. Section 4 presents algorithms and implementation details for quality improvement. We give several experimental results in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Previous Work

Hexahedral Mesh Generation The existing methods for unstructured hexahedral mesh generation can be grouped into four categories [15]: grid-based [18, 21, 22], medial surface [13], plastering [3], and whisker weaving [19]. The grid-based approaches generate a three dimensional grid of hexahedral elements in the interior of the domain. Grid-based methods are robust, but tend to generate poor quality elements near the boundary. Medial surface methods divide the whole domain into map-meshable regions by a set of medial surfaces, and then a series of templates are utilized to fill those regions. Plastering methods start with the boundaries, new hexahedra are attached to the meshing front until the volume is completely meshed. Whisker weaving builds the combinatorial dual of the mesh, then the dual mesh is converted into the primal mesh, and finally embedded into the given domain.

For multiple materials, mesh generation is a much more challenge problem. In [24], an octree-based isocontouring method [21, 22] was extended to multiple-material regions. However, the generated hexahedral meshes always have poorly shaped elements near the boundary.

Quality Improvement for Quadrilateral/Hexahedral Meshes The pillowing technique [14] was proposed to remove the cases that two neighboring elements share two edges/faces by inserting new vertices. Relocating mesh vertices is another popular approach to improve mesh quality. Laplacian smoothing [6] which relocates vertices to the arithmetic average of its neighboring vertices is simple and inexpensive, but it does not guarantee an improvement of the mesh quality and also results in degraded or inverted elements. Therefore, a number of optimization-based methods [8, 10, 11] were developed to improve the mesh quality by optimizing an objective function which reflects the element quality. Most of these improvement approaches were designed for manifold meshes. Due to the complexity of segmented meshes, quality improvement for segmented meshes is much more challenging.

Surface Smoothing Using Geometric Flows Geometric flows have been successfully used for surface modeling and designing because they are good at controlling geometric shape evolution. In the process of surface evolution, the geometric partial differential equations (PDEs) are discretized on a given mesh. On the other hand, geometric flows can also be used to fair zigzag meshes. In [4], an approach was described to fair meshes with rough features using diffusion and curvature flows. Surface diffusion flow and averaged mean curvature flow were used to smooth surface meshes in [16, 23] and [12], respectively.

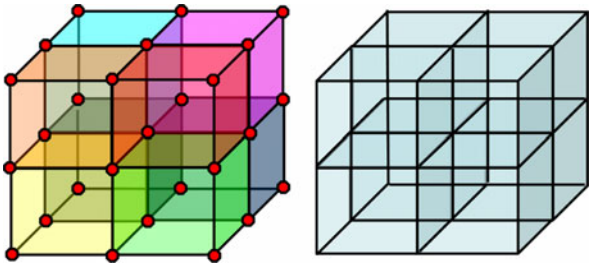
In our previous paper [12], we have proposed an geometric flow-based method for quality improvement of segmented tetrahedral meshes. The experimental results demonstrate the proposed method is effective. The generalization of improvement approach from tetrahedral meshes to hexahedral meshes is not straightforward, since a hexahedron has higher flexibility to become extremely distorted. In this paper, we will focus on quality improvement of quadrilateral/hexahedral meshes.

3 Problem Description and Preparation

In this section, we first provide the problem description of quality improvement for quadrilateral/hexahedral meshes, and then classify mesh vertices into four groups. Before introducing our mesh improvement algorithms, we should select proper quality metrics.

3.1 Problem Description

For a given mesh, quality improvement aims to make each element of the mesh has an optimal shape. A segmented hexahedral mesh for multiple regions is composed of several separated sub-meshes for each component with conforming boundaries. The union of component boundaries forms a complicated non-manifold quadrilateral mesh. Due to the complexity of segmented meshes, quality improvement is much



(a) (b)

Fig. 1 An illustration example of a multi-material domain. (**a**) is a cube consists of eight components, which are marked by different colors. (**b**) is the non-manifold lattice boundary made up of eight component boundaries

more intractable than the traditional improvement problem for single-material regions. The quadrilateral/hexahedral mesh for a segmented domain is referred as high quality, if all the elements are well shaped, and the boundary surfaces are smooth. In this paper, we intend to develop a novel geometric flow-based approach to optimize hexahedral elements in each component, and improve non-manifold quadrilateral boundary meshes simultaneously.

To simplify the non-manifold boundary, we divide the whole boundary into several surface patches sharing common boundary curves with each other. Here, the common surface shared by any two components is referred as a boundary surface patch, and the exterior boundary of each component is regarded as a boundary surface patch as well. The common curve of any two surfaces is defined as a boundary curve. As shown in Fig. 1, the cube is composed of eight small cubes representing different materials. The common faces shared by any pair of neighboring cubes are called surface patches, and black lines with red end points are boundary curves. Therefore, the boundary smoothing problem is converted to fairing and regularizing curves and surface patches.

Due to the complexity of meshes for multiple regions, we categorize mesh vertices into the following four groups:

Interior vertices: Vertices inside one volumetric component.

Surface vertices: Manifold vertices on boundary surface patches, which can move along the normal direction to smooth the surface, and can also move along the tangent direction to improve the Jacobian.

Curve vertices: Vertices located on boundary curves, which can only move along the tangent direction.

Fixed vertices: End points of boundary curves and other non-manifold vertices, which are fixed during the mesh improvement process.

Then we will handle various vertices using different algorithms. For quadrilateral meshes, there are only three types of vertices: surface vertices, curve vertices, and fixed vertices.

3.2 Quality Metrics

Various quantities have been used to measure the shape or quality of a hexahedron. Here, we choose the determinant and the condition number of Jacobian matrix [7] as quality metrics for hexahedral meshes.

Let H be a hexahedron with eight vertices $\mathbf{x}_{ijk}$ $(i, j, k = 0, 1)$, then the hexahedron can be represented as a trilinear parametric volume defined on a unit cube,

$$\mathbf{x}(u, v, w) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} u^i(1-u)^{1-i} v^j(1-v)^{1-j} w^k(1-w)^{1-k}\mathbf{x}_{ijk}. \qquad (1)$$

The Jacobian matrix

$$J(\mathbf{x}) = J(x, y, z) = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w} \end{pmatrix}$$

describes the linear transformation from the ideal shape (unit cube) to hexahedron H. If the determinant of the Jacobian matrix at all the eight vertices are positive, then the hexahedron is valid, otherwise, the hexahedron is regarded as inverted. We call the determinant of Jacobian matrix as *Jacobian*, and the determinant of the column-normalized Jacobian matrix as the *scaled Jacobian*.

The condition number of the Jacobian matrix is defined as $\kappa(J) = \frac{1}{3}\|J\|_F \times \|J^{-1}\|_F$, where $\|J\|_F = [\mathrm{tr}(J^T J)]^{1/2}$ denotes the Frobenius norm. It is easy to derive that $\kappa(J) = \frac{1}{3}\sqrt{\sum_{i,j}(\sigma_i/\sigma_j)^2} \geq 1$ is a metric with respect to the singular values $\{\sigma_i\}_{i=1}^3$ of the Jacobian matrix. The condition number reaches minimum iff $\sigma_1 = \sigma_2 = \sigma_3$.

For a quadrilateral $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$, we define the following metric

$$J(\mathbf{x}_j) = \det(\mathbf{x}_{j+1} - \mathbf{x}_j, \mathbf{x}_{j+3} - \mathbf{x}_j, \mathbf{n}_j)$$

named the *Jacobian* for each vertex, where "det" denotes determinant, the subscript of $\mathbf{x}_j$ is in module of 4, and $\mathbf{n}_j$ is the unit normal vector at vertex $\mathbf{x}_i$. The corresponding *scaled Jacobian* is $\det(\frac{\mathbf{x}_{j+1}-\mathbf{x}_j}{\|\mathbf{x}_{j+1}-\mathbf{x}_j\|}, \frac{\mathbf{x}_{j+3}-\mathbf{x}_j}{\|\mathbf{x}_{j+3}-\mathbf{x}_j\|}, \mathbf{n}_j)$.

4 Quality Improvement Algorithm and Implementation

Our quality improvement algorithm is composed of four parts: pillowing, boundary curve fairing and regularization, boundary surface fairing and regularization, and volume mesh optimization. The pillowing technique is used to remove the hexahedra with two or more faces on the boundary surface. To fair a curve/surface mesh, the vertices are relocated along its normal direction to make the curve/surface as smooth as possible. To regularize a curve/surface mesh, we intend to modify the

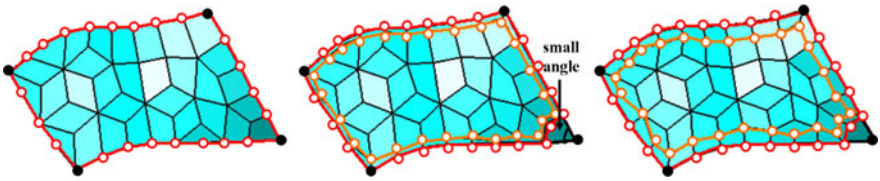


Fig. 2 Procedure of the pillowing technique applied on a surface patch. (**a**) is a given surface patch, *red curves with black end points* are boundary curves; (**b**) shows the shrink set (*red*) and a pillowed layer (*orange*); and (**c**) the inserted layer is dragged inside using the regularization technique

vertices along the tangent direction such that each quadrilateral element becomes similar to a square.

4.1 The Pillowing Technique

In a quadrilateral mesh, a doublet occurs when two elements share two edges. For non-manifold boundary, if a quadrilateral has more than two edges located on a boundary curve, we regard it as a doublet. Doublets will result in poor quality elements, and one effective method is to change the connectivity of doublet vertices. The pillowing technique can be used to improve the mesh quality by inserting one layer around the boundary curves [14]. Since the whole boundary has been divided into several manifold surface patches, mesh pillowing can be operated on each surface patch independently.

Figure 2 shows the pillowing procedure for a surface patch. First, we set the whole surface patch as a shrink set. If there is a quadrilateral with two edges forming a small angle on boundary curves (Fig. 2(b)), it would be excluded from the shrink set. The shrink set boundary is the outer layer. Second, we create a parallel layer which is a shrinkage of the outer layer. Vertex connections in the shrink set with respect to the outer layer vertices are replaced with the corresponding newly added vertices. Then, each newly added vertices is connected to its corresponding vertex on the outer layer to fill the gap between the two layers. Finally, we utilize the regularization technique introduced later to drag the inserted layer inside so as to improve the mesh quality.

For hexahedral meshes, the pillowing technique is also a popular approach to remove doublets so that any two elements have at most one common face. The pillowing idea can be generalized to eliminate the doublet that one hexahedron has two or more faces on the mesh boundary. In segmented hexahedral meshes, this kind of doublet is much more common and results in low quality elements.

We apply the pillowing algorithm (Algorithm 1) to pillow each component of the segmented meshes. Since one component shares boundary with other components, the shrink set does not shrink actually. We insert inner vertices on the pillowed layer without interfering the common boundary mesh. Figure 3 shows a simple illustration

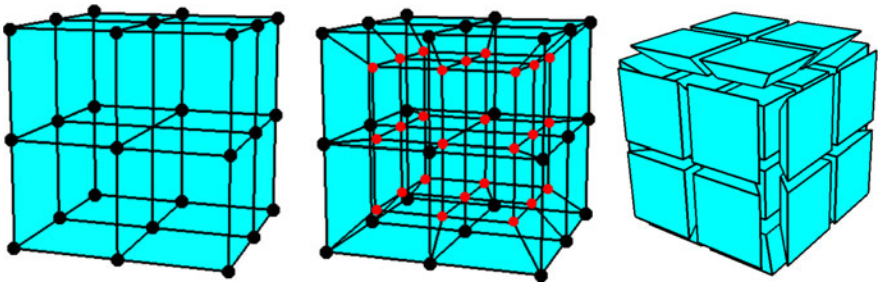


Fig. 3 An illustration of pillowing a cube component. (**a**) A hexahedral mesh of a cube; (**b**) the pillowed mesh, the *black vertices* are on the outer layer, and the *red vertices* are newly added; and (**c**) hexahedral elements after pillowing

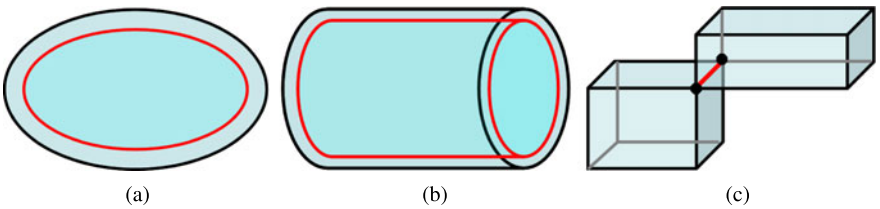


Fig. 4 (**a**) The outer layer (*black*) and the pillowed layer (*red*) of a closed component; (**b**) the outer layer (*black* one) and the pillowed layer (*red*) of an open component; and (**c**) hexahedra surrounding the non-manifold edge (*red*) should be eliminated from the shrink set

of hexahedral mesh pillowing. It can be seen that all the hexahedra have at most one face on the boundary after pillowing.

Algorithm 1 (Pillowing one component of the segmented hexahedral meshes)

1. Find the shrink set and the outer layer of the component.
   a. Set the whole component as the shrink set;
   b. For a closed component (see Fig. 4(a)), the outer layer is just the shrink set boundary; for an open component (see Fig. 4(b)), the outer layer is open as well; and
   c. Find out non-manifold boundary edges (see Fig. 4(c)), and eliminate all the hexahedra surrounding these edges from the shrink set.
2. Mesh pillowing.
   a. Create a copy of the outer layer, which is the pillowed layer. Shrink the pillowed layer along the normal direction toward the interior of the component;
   b. Loop for each hexahedron contained in the shrink set, replace the outer layer vertices by the corresponding vertices on the pillowed layer. Hence, there forms a gap between the two layers; and

c. Fill the gap by connecting each pair of opposite vertices on the two layers, and obtain several new hexahedra sandwiched between the outer layer and the pillowed layer.

3. Update the data information such as vertex type, face neighbor, and hexahedron neighbor.

4.2 Curve Smoothing Driven by Curve Diffusion Flow

Let $[\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_n]$ be a boundary curve with two fixed end points $\mathbf{x}_0$ and $\mathbf{x}_n$. To fair the curve, we introduce a temporal variable t, and evolve the curve along the normal direction at a speed with respect to curvature. Simply choosing the curvature as the speed can fair the curve but can not preserve shape features. Here, we construct a shape-preserving curve diffusion flow to evolve the curve,

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = -\big[(\Delta\boldsymbol{\kappa}_i)^T\mathbf{n}_i\big]\mathbf{n}_i, \quad i = 1,\ldots,n-1, \tag{2}$$

where

$$\boldsymbol{\kappa}_i = \frac{\mathbf{t}_{i+1}-\mathbf{t}_i}{s_i}, \qquad \mathbf{n}_i = \frac{\boldsymbol{\kappa}_i}{\|\boldsymbol{\kappa}_i\|}, \tag{3}$$

$$s_i = \frac{\|\mathbf{x}_i-\mathbf{x}_{i-1}\|+\|\mathbf{x}_i-\mathbf{x}_{i+1}\|}{2}, \qquad \mathbf{t}_i = \frac{\mathbf{x}_i-\mathbf{x}_{i-1}}{\|\mathbf{x}_i-\mathbf{x}_{i-1}\|}, \tag{4}$$

and Δ is the Laplace operator. $\mathbf{n}_i$ is a discretization of the normal direction at vertex $\mathbf{x}_i$, and $\|\boldsymbol{\kappa}_i\|$ is the corresponding curvature.

Equation (2) can be solved using the explicit Euler scheme

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} - \tau\big[(\Delta\boldsymbol{\kappa}_i)^T\mathbf{n}_i\big]\mathbf{n}_i, \quad i = 1,\ldots,n-1, \tag{5}$$

where τ is a temporal step-size, $\mathbf{x}_i^{(0)} = \mathbf{x}_i$, and $\mathbf{x}_0^{(k)} = \mathbf{x}_0^{(k+1)} = \mathbf{x}_0$, $\mathbf{x}_n^{(k)} = \mathbf{x}_n^{(k+1)} = \mathbf{x}_n$. $\boldsymbol{\kappa}_i$ and $\mathbf{n}_i$ are defined in Eq. (3) by taking $\mathbf{x}_i = \mathbf{x}_i^{(k)}$, $i = 1,\ldots,n-1$. $\Delta\boldsymbol{\kappa}_i$ is discretized as $(\frac{\boldsymbol{\kappa}_{i+1}-\boldsymbol{\kappa}_i}{\|\mathbf{x}_{i+1}-\mathbf{x}_i\|} - \frac{\boldsymbol{\kappa}_i-\boldsymbol{\kappa}_{i-1}}{\|\mathbf{x}_i-\mathbf{x}_{i-1}\|})/s_i$, with $i = 1,\ldots,n-1$, $\boldsymbol{\kappa}_0$ and $\boldsymbol{\kappa}_n$ are taken as zero vectors.

4.3 Curve Regularization

The boundary curve $[\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_n]$ is referred as regular if vertices are uniformly distributed on the curve. Therefore, it can be regularized by minimizing the following energy functional

$$\mathcal{E}(\mathcal{C}) = \frac{1}{2}\sum_{i=1}^{n}\big(\|\mathbf{x}_i-\mathbf{x}_{i-1}\|-h\big)^2, \tag{6}$$

where $h = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$ is the averaged length of each two neighboring vertices. At each free vertex $\mathbf{x}_i$ of the curve, we vary $\mathbf{x}_i$ as $\mathbf{x}_i \to \mathbf{x}_i + \varepsilon_i \boldsymbol{\Phi}_i$, $\boldsymbol{\Phi}_i \in \mathbb{R}^3$, $i = 1, \ldots, n-1$. Then we obtain the first-order variation

$$\begin{aligned}\delta(\mathcal{E}, \boldsymbol{\Phi}_i) &= \left.\frac{\partial \mathcal{E}(\mathcal{C}, \varepsilon_i)}{\partial \varepsilon_i}\right|_{\varepsilon_i = 0} \\ &= \big(\|\mathbf{x}_{i+1} - \mathbf{x}_i\| - h\big)\frac{\boldsymbol{\Phi}_i^T(\mathbf{x}_i - \mathbf{x}_{i+1})}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|} + \big(\|\mathbf{x}_i - \mathbf{x}_{i-1}\| - h\big)\frac{\boldsymbol{\Phi}_i^T(\mathbf{x}_i - \mathbf{x}_{i-1})}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|}.\end{aligned}$$

To keep the curve shape, $\boldsymbol{\Phi}_i$ is chosen as $\mathbf{e}_i$ which is the unit tangential direction at $\mathbf{x}_i$, then a set of L^2-gradient flows are derived as

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \delta(\mathcal{E}, \mathbf{e}_i)\mathbf{e}_i = \mathbf{0}, \tag{7}$$

$i = 1, \ldots, n-1$. The discretization of Eq. (7) can be written as

$$\begin{aligned}&\frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\tau} + \big(\|\mathbf{x}_{i+1}^{(k)} - \mathbf{x}_i^{(k)}\| - h\big)\frac{\mathbf{e}_i\mathbf{e}_i^T(\mathbf{x}_i^{(k)} - \mathbf{x}_{i+1}^{(k)})}{\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i+1}^{(k)}\|} \\ &\quad + \big(\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)}\| - h\big)\frac{\mathbf{e}_i\mathbf{e}_i^T(\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)})}{\|\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k)}\|} = 0.\end{aligned} \tag{8}$$

The initial value is chosen as $\mathbf{x}_i^{(0)} = \mathbf{x}_i$. Each $\mathbf{e}_i$ is calculated as the unit tangent direction of a fitting quadratic curve with respect to $\mathbf{x}_{i-1}$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$.

4.4 Surface Smoothing Using Various Geometric Flows

Geometric flows have been successfully used in surface modeling since they are inherently good at controlling geometric shape evolution. Let S_0 be a piece of compact orientable surface in $\mathbb{R}^3$ with the boundary denoted as Γ. Introducing the temporal variable t, the surface evolution can be formularized as

$$\frac{\partial \mathbf{x}(t)}{\partial t} = V_n(\mathbf{x})\mathbf{n}(\mathbf{x}), \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{9}$$

where $\mathbf{x}(t)$ is surface point located on $S(t)$, $V_n(\mathbf{x})$ denotes the normal velocity on $S(t)$ at $\mathbf{x}$, and $\mathbf{n}(\mathbf{x})$ stands for the unit normal. Since the velocity $V_n(\mathbf{x})$ usually represents several geometric quantities which reflect geometric properties of the evolving surface, Eq. (9) is referred as a geometric flow.

Various geometric flows can be constructed to meet different application requirements by choosing an appropriate normal velocity $V_n(\mathbf{x})$. Curvature is an important descriptor reflecting the flexibility of surface, hence geometric PDEs are basically expressed by curvatures. The most common used geometric flows include Mean Curvature Flow (MCF), Averaged Mean Curvature Flow (AMCF), Surface Diffusion Flow (SDF) and Willmore Flow (WF). MCF can be used to get the minimum

surface with respect to fixed boundary. AMCF and SDF are volume-preserving during the evolution. At first, we would like to introduce definitions of these four well-used geometric flows [17, 20].

Definition 1 (Mean curvature flow (MCF))

$$\frac{\partial \mathbf{x}}{\partial t} = 2\mathbf{H}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{10}$$

where $\mathbf{H}$ denotes the mean curvature vector. MCF is an area-reducing flow, which can be used to get the minimum surface with respect to a fixed boundary.

Definition 2 (Averaged mean curvature flow (AMCF))

$$\frac{\partial \mathbf{x}}{\partial t} = \big[H - h(t)\big]\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{11}$$

where

$$h(t) = \int_S H \,\mathrm{d}A \Big/ \int_S \mathrm{d}A.$$

Since $h(t)$ is the average of the mean curvature H on the whole surface, hence Eq. (11) is called the averaged mean curvature flow [5].

Definition 3 (Surface diffusion flow (SDF))

$$\frac{\partial \mathbf{x}}{\partial t} = -2\Delta_s H\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{12}$$

where Δ_s is the Laplace–Beltrami operator. It is an area-reducing and volume-preserving flow which can be used for noise removing in surface design.

Definition 4 (Willmore flow (WF))

$$\frac{\partial \mathbf{x}}{\partial t} = -\big[\Delta_s H + 2H\big(H^2 - K\big)\big]\mathbf{n}, \qquad S(0) = S_0, \qquad \partial S(t) = \Gamma, \tag{13}$$

where H and K are the mean curvature and Gaussian curvature, respectively. WF has been investigated and used widely in computational geometry and other fields. Suppose the initial surface is a sphere, WF can keep the sphere without evolution.

The following theorems [20] describe area variation and volume variation of the evolving surface, respectively.

Theorem 1 *Let $V(t)$ denote the (directional) volume of the region enclosed by $S(0)$ and $S(t)$ (see Fig. 5 for a 2D curve case). Then we have*

$$\frac{dV(t)}{dt} = \int_{S(t)} V_n(\mathbf{x}) \,\mathrm{d}A. \tag{14}$$

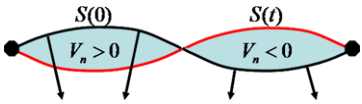


Fig. 5 The directional area between the curves $S(0)$ and $S(t)$. The area of the region with normal velocity $V_n > 0$ (or $V_n < 0$)

Taking $V_n = H(t) - h(t)$, where $h(t) = \int_{S(t)} H \, \mathrm{d}A / \int_{S(t)} \mathrm{d}A$, then we have

$$\frac{dV(t)}{\mathrm{d}t} = \int_{S(t)} \big(H(\mathbf{x}) - h(t)\big) \, \mathrm{d}t = \int_{S(t)} H \, \mathrm{d}A - h(t) \int_{S(t)} \mathrm{d}A = 0.$$

Hence AMCF is volume-preserving. Similarly, with $V_n = -2\Delta_s H$,

$$\frac{\mathrm{d}}{\mathrm{d}t} V(t) = -\frac{2}{3} \int \mathrm{div}_s(\nabla_s H) \, \mathrm{d}A = \frac{2}{3} \int (\nabla_s H)^{\mathrm{T}} \nabla_s(1) \, \mathrm{d}A = 0,$$

where div_s and ∇_s are the tangential divergence operator and the tangential gradient operator, respectively. Thus, SDF is volume-preserving as well.

Theorem 2 *Let $A(t)$ be the are $S(t)$, then we have*

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t} = -\int_{S(t)} V_n(\mathbf{x})^{\mathrm{T}} \mathbf{H} \, \mathrm{d}A. \tag{15}$$

For MCF, we have

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t} = -2 \int_{S(t)} H^2 \, \mathrm{d}A < 0,$$

which means the surface area keeps reducing until the mean curvature $H = 0$ all over the surface. Hence, the steady solution depends upon the fixed boundary curves, while the enclosed surface will shrink to a point. SDF is another area-reducing flow, unlike MCF, SDF decreases the surface area until H is constant. WF is a gradient flow corresponding to the Willmore energy [1, 2]

$$E(S) = \int_S H^2 \, \mathrm{d}A,$$

which evolves the surface S by decreasing the Willmore energy at the steepest direction. The Willmore energy is a scale invariant. For any sphere, the Willmore energy is 4π, and the sphere is a global minimum for an enclosed surface.

All the above four geometric flows will be applied for surface fairing. Since the geometric flows evolve surface within a pre-defined range, the initial features will not be destroyed seriously. Let S be a quadrilateral surface patch and $\{\mathbf{x}_i\}_{i=1}^N$ be its free vertex set. For a vertex $\mathbf{x}_i$ with valence $2n_i$, $N(i) = \{i_1, i_2, \ldots, i_{n_i}, i'_1, i'_2, \ldots, i'_{n_i}\}$ denotes the index set of the first-ring neighbors of $\mathbf{x}_i$. Geometric PDEs are solved on the quadrilateral mesh S using an explicit discretization method, where the discrete approximation of the mean curvature vector, mean curvature, Gaussian curvature, and surface normal are required. These approximations can be obtained from the quadratic fitting surface with respect to $\mathbf{x}_i$ and its first-ring neighbors [20].

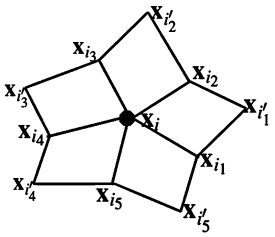


Fig. 6 The first ring neighborhood of $\mathbf{x}_i$

Discretization of Geometric PDEs An Euler explicit discrete scheme

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}}{\tau}$$

is used in the temporal direction. In the averaged mean curvature flow, $h(t)$ can be discretized as $h(t) = \int_{S(t)} H\,\mathrm{d}A / \int_{S(t)} \mathrm{d}A = \sum_{i=1}^{N} [H(\mathbf{x}_i) A_{S(t)}(\mathbf{x}_i)] / A(S(t))$, $A(S(t))$ is the total area of the quadrilateral mesh $S(t)$, $A_{S(t)}(\mathbf{x}_i)$ is one fourth of the first ring neighbor area surrounding vertex $\mathbf{x}_i$, where the first ring neighborhood of $\mathbf{x}_i$ is shown in Fig. 6.

Next, we compute the mean curvature $H(\mathbf{x}_i)$, the Gaussian curvature $K(\mathbf{x}_i)$, and the Laplace–Beltrami operator Δ_s at vertex $\mathbf{x}_i$ on quadrilateral meshes. Suppose the vertex $\mathbf{x}_i$ has a valence of n, and its neighboring vertices are $\mathbf{x}_{i_j}$ $(i_j \in N(i))$. First, we fit $\mathbf{x}_i$ and its neighboring vertices to a quadric surface in the local coordinate system via the algorithm proposed in [20]. The basis function is chosen as

$$\{B_l(u,v)\}_{l=0}^{5} = \left\{1, u, v, \frac{1}{2}u^2, uv, \frac{1}{2}v^2\right\},$$

then the problem is to determine coefficients $\mathbf{c}_l \in \mathbb{R}^3$ for the parametric-form fitted surface $\mathbf{x}(u,v) := \sum_{l=0}^{5} \mathbf{c}_l B_l(u,v)$, such that

$$\sum_{l=0}^{5} \mathbf{c}_l B_l(\mathbf{q}_k) = \mathbf{x}_{i_k}, \quad k = 0, \ldots, n$$

in the least square sense. Here i_0 is denoted as i, and $\mathbf{q}_k$ is the local coordinate of $\mathbf{x}_{i_k}$ on the tangent plane of $\mathbf{x}_i$. After determining $\{\mathbf{c}_l\}_{l=0}^{5}$, it is easy to compute $\mathbf{x}_u$, $\mathbf{x}_v$, $g_{11} = \langle \mathbf{x}_u, \mathbf{x}_u \rangle$, $g_{12} = \langle \mathbf{x}_u, \mathbf{x}_v \rangle$, $g_{22} = \langle \mathbf{x}_v, \mathbf{x}_v \rangle$, $g = g_{11}g_{22} - g_{12}g_{12}$, $b_{11} = \langle \mathbf{x}_{uu}, \mathbf{n} \rangle$, $b_{12} = \langle \mathbf{x}_{uv}, \mathbf{n} \rangle$, $b_{22} = \langle \mathbf{x}_{vv}, \mathbf{n} \rangle$, $g_{\alpha\beta\gamma} = \langle \mathbf{x}_{u^\alpha}, \mathbf{x}_{u^\beta u^\gamma} \rangle$ $(\alpha, \beta, \gamma = 1, 2)$, and $\mathbf{x}_{u^\alpha u^\beta} = \frac{\partial^2 \mathbf{x}}{\partial u^\alpha \partial u^\beta}$ $(\alpha, \beta = 1, 2)$.

Using the approximate equation given in [20], we can calculate the mean curvature vector, the Gaussian curvature, and the Laplace–Beltrami operator as follows.

$$H(\mathbf{x}_i)\mathbf{n} = \mathbf{H}(\mathbf{x}_i) = \frac{1}{2}\Delta_s \mathbf{x}_i \approx \frac{1}{2}\sum_{j=0}^{n} w_{ij}^{\Delta} \mathbf{x}_{i_j},$$

where we use the superscript "Δ" to denote the approximation coefficient for the Laplacian–Beltrami operator Δ_s.

$$
\begin{aligned}
w_{i,j}^{\Delta} &= g_u^{\Delta} c_1^{(j)} + g_v^{\Delta} c_2^{(j)} + g_{uu}^{\Delta} c_3^{(j)} + g_{uv}^{\Delta} c_4^{(j)} + g_{vv}^{\Delta} c_5^{(j)},\\
g_u^{\Delta} &= -\big[g_{11}(g_{22}g_{122} - g_{12}g_{222}) + 2g_{12}(g_{12}g_{212} - g_{22}g_{112})\\
&\quad + g_{22}(g_{22}g_{111} - g_{12}g_{211})\big]/g^2,\\
g_v^{\Delta} &= -\big[g_{11}(g_{11}g_{222} - g_{12}g_{122}) + 2g_{12}(g_{12}g_{112} - g_{11}g_{212})\\
&\quad + g_{22}(g_{11}g_{211} - g_{12}g_{111})\big]/g^2,\\
g_{uu}^{\Delta} &= \frac{g_{22}}{g}, \qquad g_{uv}^{\Delta} = -\frac{2g_{12}}{g}, \qquad g_{vv}^{\Delta} = \frac{g_{11}}{g},
\end{aligned}
$$

and $c_l^{(j)}$ ($l = 1, \ldots, 5$, $j = 0, \ldots, n$) is the $(l+1, j+1)$-th element of $\mathbf{C}$.

For the Gaussian curvature, we have

$$
K(\mathbf{x}_i)\mathbf{n} = \mathbf{K}(\mathbf{x}_i) = \frac{1}{2}\Box\mathbf{x}_i \approx \frac{1}{2}\sum_{j=1}^{n} w_{ij}^{\Box}\mathbf{x}_{i_j},
$$

where "$\Box$" denotes the Giaquinta–Hildebrandt operator,

$$
\begin{aligned}
w_{i,j}^{\Box} &= g_u^{\Box} c_1^{(j)} + g_v^{\Box} c_2^{(j)} + g_{uu}^{\Box} c_{11}^{(j)} + g_{uv}^{\Box} c_{12}^{(j)} + g_{vv}^{\Box} c_{22}^{(j)},\\
g_u^{\Box} &= -\big[b_{11}(g_{22}g_{122} - g_{12}g_{222}) + 2b_{12}(g_{12}g_{212} - g_{22}g_{112})\\
&\quad + b_{22}(g_{22}g_{111} - g_{12}g_{211})\big]/g^2,\\
g_v^{\Box} &= -\big[b_{11}(g_{11}g_{222} - g_{12}g_{122}) + 2b_{12}(g_{12}g_{112} - g_{11}g_{212})\\
&\quad + b_{22}(g_{11}g_{211} - g_{12}g_{111})\big]/g^2,\\
g_{uu}^{\Box} &= \frac{b_{22}}{g}, \qquad g_{uv}^{\Box} = -\frac{2b_{12}}{g}, \quad \text{and} \quad g_{vv}^{\Box} = \frac{b_{11}}{g}.
\end{aligned}
$$

4.5 Regularization of Boundary Quadrilateral Mesh

Generally, a quadrilateral mesh is referred as regular if its vertices are equally distributed, and each quadrilateral is close to a square. For a given quadrilateral surface patch $\mathcal{S}$, we use the following energy functional to describe its regularity,

$$
\mathcal{E}(\mathcal{S}) = \frac{1}{2}\sum_{i=1}^{N}\big(E_1(\mathbf{x}_i) + \lambda_1 E_2(\mathbf{x}_i) + \lambda_2 E_3(\mathbf{x}_i)\big), \tag{16}
$$

where

$$
\begin{aligned}
E_1(\mathbf{x}_i) &= \sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)^2,\\
E_2(\mathbf{x}_i) &= \sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big)^2,
\end{aligned} \tag{17}
$$

$$E_3(\mathbf{x}_i) = \sum_{j=1}^{n_i} \big(\det(\mathbf{x}_{i_j} - \mathbf{x}_i, \mathbf{x}_{i_{j+1}} - \mathbf{x}_i, \mathbf{n}_i) - J_i\big)^2.$$

In Eq. (16), $\{\mathbf{x}_i\}_{i=1}^N$ are free vertices of surface $\mathcal{S}$. For the vertex $\mathbf{x}_i$ with the quadrilateral valence of n_i, let $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{n_i}}$ be the neighboring vertices connected with $\mathbf{x}_i$, and $\mathbf{x}_{i'_1}, \ldots, \mathbf{x}_{i'_{n_i}}$ be the opposite vertices of $\mathbf{x}_i$ in the quadrilateral $[\mathbf{x}_i \mathbf{x}_{i_j} \mathbf{x}_{i'_j} \mathbf{x}_{i_{j+1}}]$ $(j = 1, \ldots, n_i)$, $n_i + 1 \triangleq 1$. Figure 6 illustrates the case with $n_i = 5$. We intend to regularize the quadrilateral mesh by minimizing the energy functional (16) which is the combination of three terms:

(1) Obviously, $\sum_{i=1}^N E_1(\mathbf{x}_i)$ is globally minimized when the distance between each pair of neighboring vertices equals to h, where $h = \sqrt{A_m}$, and A_m is the average area of all the quadrilaterals.
(2) $\sum_{i=1}^N E_2(\mathbf{x}_i)$ is used to force the diagonals of each quadrilateral as long as $\sqrt{2}h$, so as to avoid the existence of slender elements.
(3) In the third term $E_3(\mathbf{x}_i)$, J_i stands for the averaged Jacobian with respect to $\mathbf{x}_i$, and $\mathbf{n}_i$ is the unit normal direction at $\mathbf{x}_i$.

At each free vertex $\mathbf{x}_i$, we vary $\mathbf{x}_i$ as $\mathbf{x}_i \to \mathbf{x}_i + \varepsilon_i \Phi_i$, where $\Phi_i \in \mathbb{R}^3$, $i = 1, \ldots, N$. It is easy to derive the following first order variation form

$$\begin{aligned}\delta\big(\mathcal{E}(\mathcal{S}), \Phi_i\big) = &\sum_{j=1}^{n_i} \big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big) \frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i_j})}{\|\mathbf{x}_i - \mathbf{x}_{i_j}\|} \\ &+ \lambda_1 \sum_{j=1}^{n_i} \big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big) \frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i'_j})}{\|\mathbf{x}_i - \mathbf{x}_{i'_j}\|} \\ &+ \lambda_2 \sum_{j=1}^{n_i} \big(\det(\mathbf{x}_{i_j} - \mathbf{x}_i, \mathbf{x}_{i_{j+1}} - \mathbf{x}_i, \mathbf{n}_i) - J_i\big) \det(\Phi_i, \mathbf{x}_{i_j} - \mathbf{x}_{i_{j+1}}, \mathbf{n}_i).\end{aligned}$$

To preserve the surface shape, all the free vertices are forced to move on its tangential plane. Let $\mathbf{e}_i^{(1)}$ and $\mathbf{e}_i^{(2)}$ be two unit orthogonal tangential directions at $\mathbf{x}_i$, we construct the following two sets of L^2-gradient flows from the first-order variations,

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \delta\big(\mathcal{E}(\mathcal{S}), \mathbf{e}_i^{(l)}\big)\mathbf{e}_i^{(l)} = 0, \quad l = 1, 2. \tag{18}$$

An explicit Euler scheme is applied to solve the L^2-gradient flows with unknown $\mathbf{x}_i$, $i = 1, \ldots, N$.

Remark 1 In the energy functional (16), h is global. In practice, the local $h_i = \sqrt{A_i}$ can be used for each $\mathbf{x}_i$ as well, where A_i is the average area of quadrilaterals surrounding $\mathbf{x}_i$. During the iteration process, either the global h or the local h_i should be updated.

4.6 Hexahedral Mesh Optimization

For hexahedral mesh optimization, the above algorithms can be used to improve boundary quadrilateral meshes. Here, we introduce three approaches to optimize the shape of hexahedra elements by modifying the interior vertices.

4.6.1 Local Optimization

During the process of curve fairing and surface smoothing, elements nearby curves and surfaces maybe inverted. Here, we use a simple and fast local optimization approach proposed in [8] to untangle the hexahedral mesh. The vertex with negative Jacobian is relocated such that

$$\max \min_{j=1,\ldots,n_i} Jacobian_j(\mathbf{x}_i), \tag{19}$$

where $Jacobian_j(\mathbf{x}_i)$ denotes the Jacobian of $\mathbf{x}_i$ with respect to its j-th neighboring hexahedron, n_i is the vertex valence of $\mathbf{x}_i$. The optimization problem (19) is a linear programming problem which can be solved by the simplex method.

4.6.2 Global Optimization

Suppose $\{\mathbf{x}_i\}_{i=1}^N$ is the set of all interior vertices in a hexahedral mesh, for each $\mathbf{x}_i$, n_i, n_i', and n_i'' are the vertex valence, quadrilateral valence, and hexahedral valence, respectively. To optimize the whole quality of the hexahedral mesh, we minimize the following energy functional,

$$\mathcal{E}(\mathcal{H}) = \frac{1}{2}\sum_{i=1}^{N} E_1(\mathbf{x}_i) + \lambda \sum_{i=1}^{N} E_2(\mathbf{x}_i), \tag{20}$$

where

$$\begin{aligned} E_1(\mathbf{x}_i) = &\sum_{j=1}^{n_i}\left(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\right)^2 + \sum_{j=1}^{n_i'}\left(\|\mathbf{x}_{i_j'} - \mathbf{x}_i\| - \sqrt{2}h\right)^2 \\ &+ \sum_{j=1}^{n_i''}\left(\|\mathbf{x}_{i_j''} - \mathbf{x}_i\| - \sqrt{3}h\right)^2, \\ E_2(\mathbf{x}_i) = &\sum_{j=1}^{n_i''}\left(\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i) - J_i\right)^2. \end{aligned} \tag{21}$$

In Eq. (21), $h = \sqrt[3]{A}$, and A is the average volume of hexahedra in one component. J_i stands for the averaged Jacobian with respect to $\mathbf{x}_i$. $\{\mathbf{x}_{i_j}\}_{j=1}^{n_i}$ is the neighboring vertices connected with $\mathbf{x}_i$, $\{\mathbf{x}_{i_j'}\}_{j=1}^{n_i'}$ are opposite vertices of each neighboring

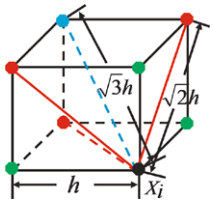


Fig. 7 Neighboring vertices of $\mathbf{x}_i$ in a hexahedron. *Green points* are neighboring vertices connected with $\mathbf{x}_i$; *red points* are opposite vertices of neighboring quadrilaterals; the *blue point* is the diagonal vertex of $\mathbf{x}_i$ in the hexahedron. Distances between neighboring vertices and $\mathbf{x}_i$ are expected to be h, $\sqrt{2}h$, and $\sqrt{3}h$, respectively

quadrilateral, and $\{\mathbf{x}_{i''_j}\}_{j=1}^{n''_i}$ are opposite vertices of each neighboring hexahedron. $\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i)$ is the Jacobian of $\mathbf{x}_i$ with respect to its j-th neighboring hexahedron, and $\mathbf{x}_{i_{j_1}}, \mathbf{x}_{i_{j_2}}, \mathbf{x}_{i_{j_3}}$ are the three neighboring vertices connected with $\mathbf{x}_i$ in the hexahedron.

The first term of the energy functional attempts to make the vertex distance in each hexahedron satisfy the relationship as shown in Fig. 7. The second term intends to make the Jacobians of $\mathbf{x}_i$ equal to the averaged Jacobian J_i. We can derive the first order variation of the energy functional (20) as follows:

$$\begin{aligned}
\delta\big(\mathcal{E}(\mathcal{H}), \Phi_i\big) = &\sum_{j=1}^{n_i}\big(\|\mathbf{x}_{i_j} - \mathbf{x}_i\| - h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i_j})}{\|\mathbf{x}_i - \mathbf{x}_{i_j}\|} \\
&+ \sum_{j=1}^{n'_i}\big(\|\mathbf{x}_{i'_j} - \mathbf{x}_i\| - \sqrt{2}h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i'_j})}{\|\mathbf{x}_i - \mathbf{x}_{i'_j}\|} \\
&+ \sum_{j=1}^{n''_i}\big(\|\mathbf{x}_{i''_j} - \mathbf{x}_i\| - \sqrt{3}h\big)\frac{\Phi_i^T(\mathbf{x}_i - \mathbf{x}_{i''_j})}{\|\mathbf{x}_i - \mathbf{x}_{i''_j}\|} \\
&- \lambda\sum_{j=1}^{n''_i}\big(\det(\mathbf{x}_{i_{j_1}} - \mathbf{x}_i, \mathbf{x}_{i_{j_2}} - \mathbf{x}_i, \mathbf{x}_{i_{j_3}} - \mathbf{x}_i) - J_i\big) \\
&\times \Phi_i^T\big((\mathbf{x}_{i_{j_2}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_3}} - \mathbf{x}_i)\big) + \big((\mathbf{x}_{i_{j_3}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_1}} - \mathbf{x}_i)\big) \\
&+ \big((\mathbf{x}_{i_{j_1}} - \mathbf{x}_i) \times (\mathbf{x}_{i_{j_2}} - \mathbf{x}_i)\big).
\end{aligned}$$

Then we move each interior vertices using the L^2-gradient flow:

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} + \sum_{l=1}^{3}\delta\big(\mathcal{E}(\mathcal{S}), \mathbf{e}^{(l)}\big)\mathbf{e}^{(l)} = 0. \tag{22}$$

Where $\mathbf{e}^{(1)} = (1, 0, 0)^T$, $\mathbf{e}^{(2)} = (0, 1, 0)^T$, $\mathbf{e}^{(3)} = (0, 0, 1)^T$. The equation is solved using explicit Euler scheme with unknown interior vertices.

The global optimization method has the advantage of improving the whole mesh quality, but it cannot guarantee all the hexahedra are valid. Thus, we combine the global optimization with the local optimization in our mesh improvement algorithm.

4.6.3 Further Improvement

Generally, after local and global optimization, we can get high quality hexahedral meshes with no degraded or inverted elements. However, for the meshes with complicated boundaries, there still exists several negative Jacobians for boundary vertices, since the above two optimization approaches only optimize the interior vertex Jacobians. Hence, we intend to modify neighboring interior vertices of those boundary vertices to eliminate negative Jacobians.

The concrete procedure has three steps. First, we loop for all the hexahedra and compute eight Jacobians for each vertex. Second, loop for each vertex, if the vertex has any Jacobian less than a given threshold, then move its neighboring interior vertices along the gradient direction of Jacobian to increase the Jacobian. Third, gradually increase the threshold, and repeat the previous two steps.

5 Application Examples and Discussion

In this section, we choose one biological dataset and two microstructure datasets to demonstrate the effectiveness of the proposed quality improvement method. For each of dataset, two meshes were generated by an octree-based method [24]: one is the boundary quadrilateral mesh, and the other is the hexahedral mesh. In the following, we will show the improvement results for these meshes.

5.1 Surface Smoothing Using Various Geometric Flows

In Sect. 4.4, we introduced four typical geometric flows: mean curvature flow (MCF), averaged mean curvature flow (AMCF), surface diffusion flow (SDF), and Willmore flow (WF). These geometric flows have their own specific properties, and can be used in different applications. In some practical applications, it is prerequested that the object volume, the boundary area, or the shape features should be preserved. All the four geometric flows can be applied for surface smoothing. To compare the smoothing effects, we validate the four geometric flows on a biological mesh dataset named ATcpnα, which is a chaperonin subunit of an archaea Acidianus tengchongensis strain S5T.

As shown in Fig. 8(a) and Fig. 9(a), the original quadrilateral mesh consists of 103,746 vertices and 104,366 quadrilaterals. Before surface smoothing, vertices are modified along the tangent directions to get a relatively regular mesh, since geometric PDEs discretized on irregular meshes always result in numerical error or

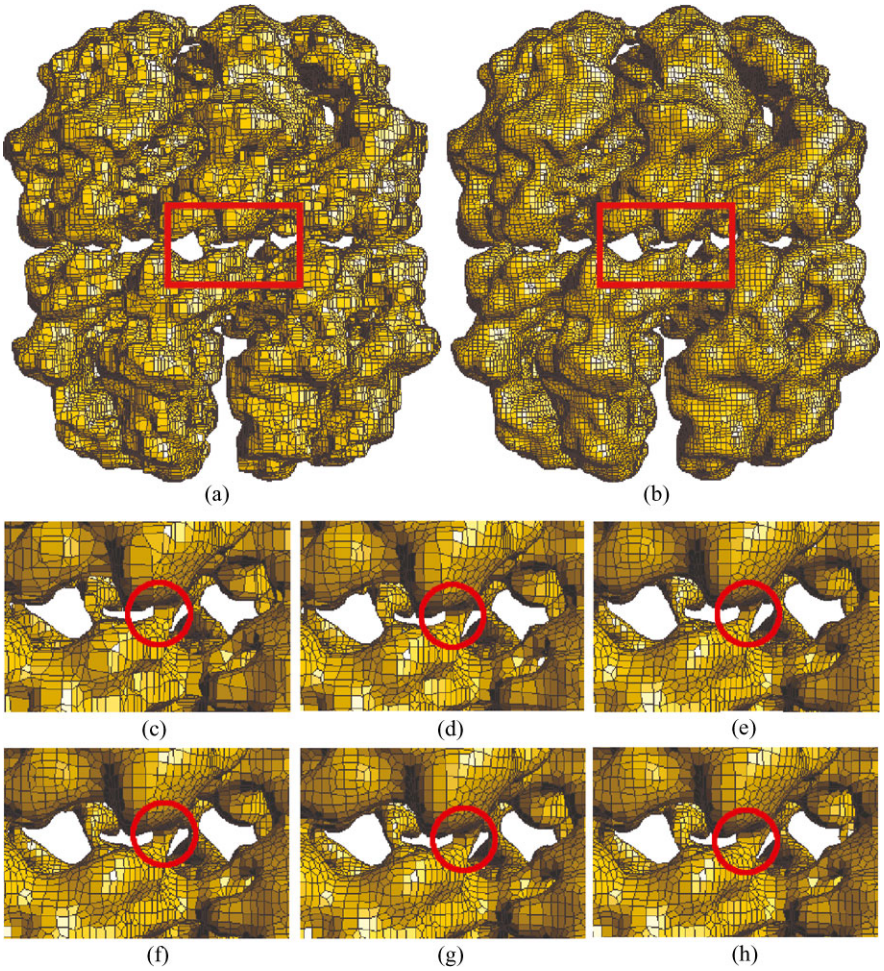


Fig. 8 Quadrilateral mesh of ATcpnα (side view). (**a**) The original mesh; (**b**) the smoothed mesh using surface diffusion flow; (**c**) the enlargement for the red window in (**a**); (**d**) the regularized mesh; and (**e**)–(**h**) are smoothed results using MCF, AMCF, SDF and WF, respectively

even divergence. By minimizing the energy functional (16), we obtain a regularized mesh with well-shaped elements, and the statistics of Jocabians are given in Table 1. Then, MCF, AMCF, SDF, and WF are applied to denoise the regularized but bumpy quadrilateral mesh.

For these four geometric flows, we choose the same temporal step size and iteration number. The smoothing process has 400 iterations, and vertices are re-regularized along the tangent directions for every 100 iterations. The total area of quadrilateral meshes for each iterative step is plotted in Fig. 10. The quality statistics of mesh smoothing by various geometric flows are given in Table 1. Moreover, the volume enclosed by the meshes are calculated to investigate the volume-preserving

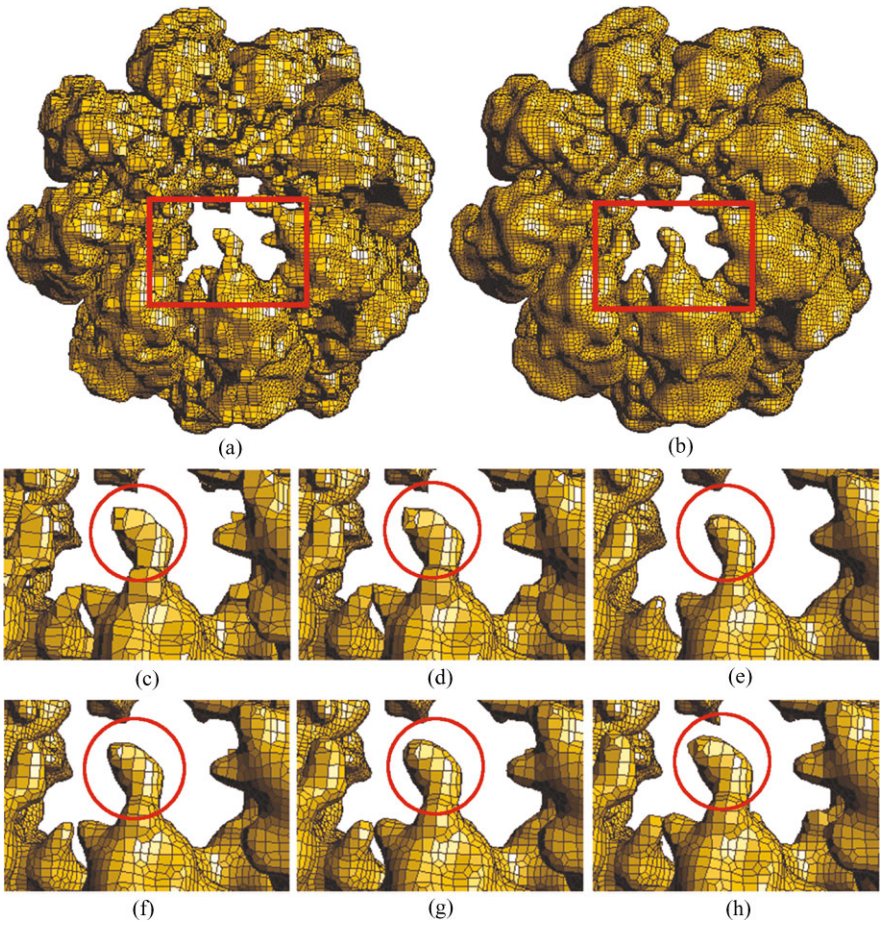


Fig. 9 Quadrilateral mesh of ATcpnα (top view). (**a**) The original mesh; (**b**) the smoothed mesh using surface diffusion flow; (**c**) the enlargement for the red window in (**a**); (**d**) the regularized mesh; and (**e**)–(**h**) are smoothed results using MCF, AMCF, SDF and WF, respectively

Table 1 Mesh quality comparison for using different geometric flows

Mesh	Jacobian		Number of Jacobian						Volume
	Worst	Best	Negative	0.0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1.0	
Original	−0.9798	1.0000	1,214	2,175	7,242	19,922	62,442	324,469	143309.9
Regularized	0.0324	1.0000	0	147	1,288	6,626	47,657	361,746	143357.0
MCF	−0.3490	1.0000	9	165	1,244	5,000	35,989	375,057	140378.1 (2.07%)
AMCF	0.0105	1.0000	0	18	815	4,082	34,459	378,090	143344.6 (0.01%)
SDF	0.0561	1.0000	0	81	735	3,841	32,497	380,310	143690.9 (0.23%)
WF	0.0767	1.0000	0	87	706	3,993	33,891	378,787	143393.0 (0.02%)

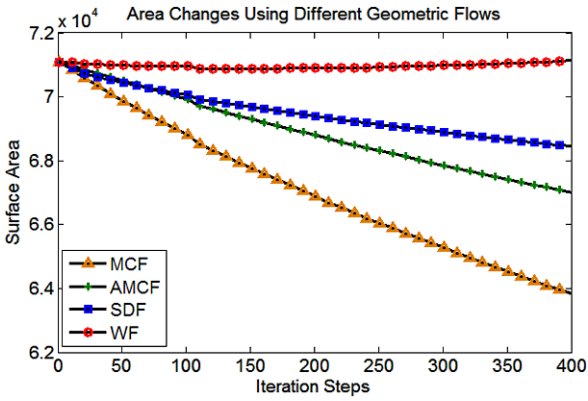


Fig. 10 Surface area changes during the evolution driven by four geometric flows

property of those geometric flows. Except for MCF, the other three geometric flows keep the volume well (the volume change is within 0.3%).

The MCF aims to evolve the surface along the normal direction at the speed of the mean curvature, which is simple and intuitive. From the definition of MCF (10), the evolution stops when $H = 0$ all over the surface. Therefore, the enclosed surface will shrink to a point eventually. Moreover, MCF can be used to get the minimum surface according to the given boundary curve. In Fig. 10, it is clear that the MCF reduces the surface area at the fastest speed among the four flows. As shown in Fig. 8(e) and Fig. 9(e), the bumpy surface can be well-smoothed using the MCF, but also along with the inevitable shrinkage.

AMCF and SDF are two volume-preserving flows. AMCF intends to equalize the mean curvature all over the surface, which seems unreasonable for a complicated surface. As a fourth order geometric flow, SDF takes account of the 1-ring and 2-ring neighbor vertices, and intends to make the mean curvature vary gradually. In Fig. 10, it can be seen that, AMCF decreases the surface area slower than MCF but faster than SDF.

WF has the property of driving a surface to a sphere, no matter how small the neck is, and the terminate sphere radius depends on the initial surface. Figure 8(h) shows the tiny expansion of thin necks. Theoretically, WF is not area-preserving and volume-preserving. However, in this example, WF almost keeps the surface area (see Fig. 10) and the enclosed volume (see Table 1).

Comparing the resulting meshes in Fig. 8 and Fig. 9, we discovered that the SDF preserves surface feature better than the other three flows. In the process of mesh smoothing, the area reduction is reasonable since the given mesh is bumpy. In the following application examples, we choose the SDF to evolve the boundary surfaces.

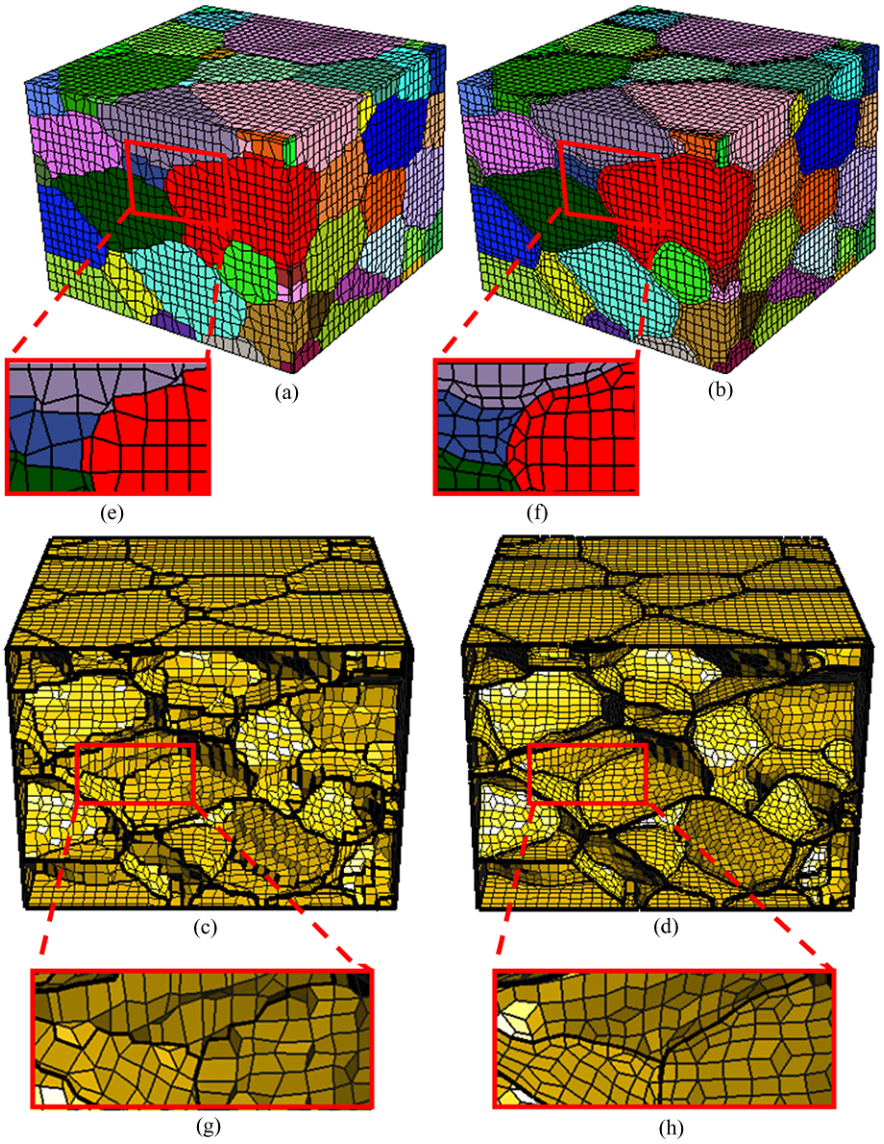


Fig. 11 92-grain microstructure. (**a**) The exterior of the original mesh; (**c**) the exterior of the improved mesh; (**d**) the interior of the original mesh; (**f**) the interior of the improved mesh; and (**e**)–(**h**) show the enlargement of *red windows* in (**a**)–(**d**), respectively

5.2 Quality Improvement for Quadrilateral Meshes

The proposed approach is then applied to two titanium alloy microstructure datasets. The two datasets are composed of 92 grains (see Fig. 11) and 52 grains (see Fig. 12),

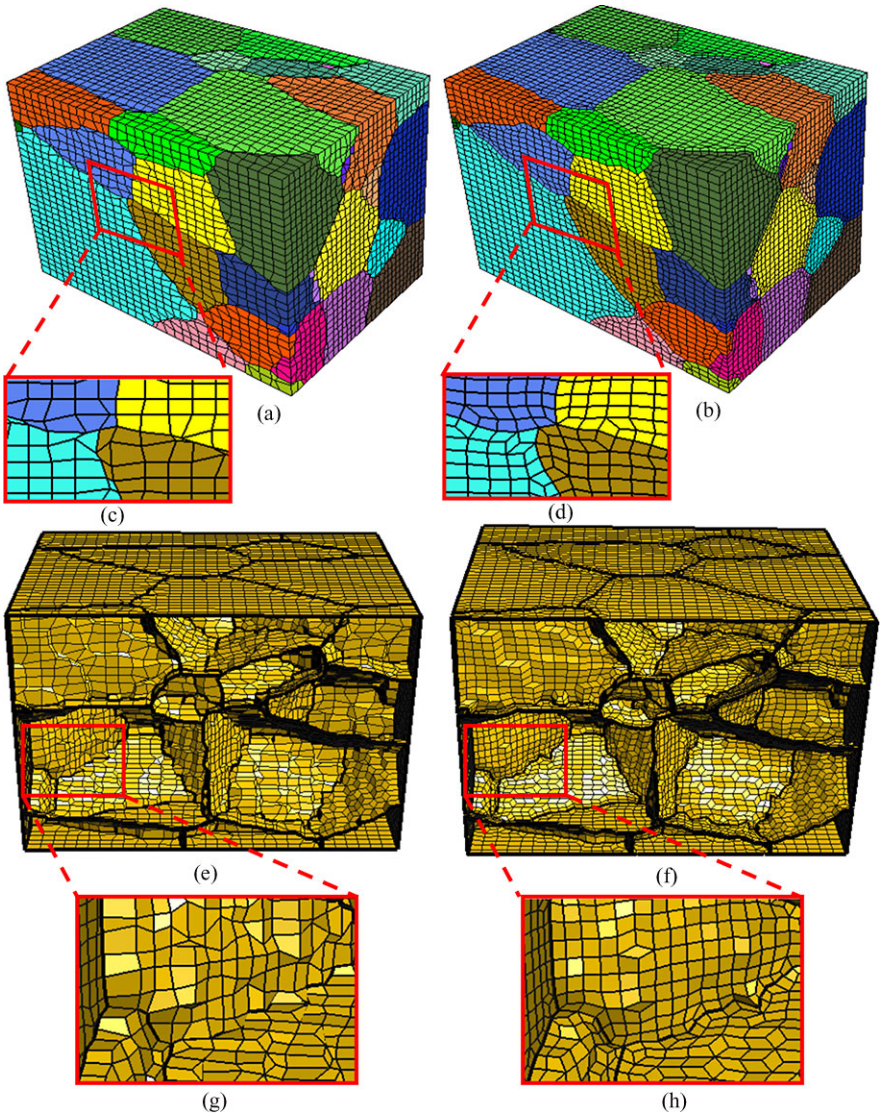


Fig. 12 52-grain microstructure. (**a**) The exterior of the original mesh; (**c**) the exterior of the improved mesh; (**d**) the interior of the original mesh; (**f**) the interior of the improved mesh; and (**e**)–(**h**) show the enlargement of *red windows* in (**a**)–(**d**), respectively

respectively. The union of all the grain boundaries forms a non-manifold boundary. The given quadrilateral meshes of the two non-manifold boundaries are given in Fig. 11(a) and Fig. 12(a). There are a great number of poorly-shaped quadrilaterals in the original meshes. Mesh vertices are irregularly distributed, the boundary curves and surfaces are bumpy.

Table 2 Quality comparison of quadrilateral meshes before and after improvement

	Mesh	Mesh size (vertex, quad)	Jacobian		Number of Jacobian					
			Worst	Best	Negative	0.0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1.0
92-grain	Original	(13,690, 15,459)	−0.8711	1.0000	151	336	965	2,855	7,966	49,561
	Improved	(24,258, 26,027)	0.1052	1.0000	0	21	323	2,426	9,963	91,375
52-grain	Original	(13,511, 14,738)	−0.6129	1.0000	150	397	1,082	2,520	7,778	47,023
	Improved	(20,823, 22,050)	0.1221	1.0000	0	26	258	2,053	11,461	74,402

To improve the quadrilateral meshes, we first divide the mesh into several manifold surface patches and boundary curves. Since the outline of the two data volumes is a cuboid, we treat the eight corners as fixed vertices, and cuboid edges as boundary curves. Then the algorithms presented in Sects. 4.2–4.5 are applied to the boundary curves and surface patches. Since there are several poor quality quadrilaterals with more than two edges on the boundary curve, the pillowing technique is used to eliminate these cases by inserting some vertices.

After quality improvement, we obtain remarkable optimized quadrilateral meshes. Fig. 11 and Fig. 12 show the contrast between meshes before and after quality improvement. It is obvious that both curves and surfaces in the improved meshes are smooth. Moreover, the vertices are uniformly distributed with no poorly shaped quadrilaterals. Table 2 lists the statistics of the scaled Jacobian for the two meshes. As shown in the table, there are a great number of negative Jacobians in the original mesh. Our improvement method makes all the Jacobian greater than 0.1, the overall mesh quality is significantly upgraded with the number of good elements (Jacobian > 0.6) increased and the number of poor elements (Jacobian < 0.4) reduced.

5.3 Quality Improvement for Hexahedral Meshes

We further validate the proposed improvement method on the hexahedral meshes of the three datasets: ATcpnα, 92-grain, and 52-grain titanium alloy microstructure. Approaches proposed in Sects. 4.2–4.5 are applied to smooth and regularize boundary meshes. Since there are several hexahedra with more than one face on the boundary, mesh pillowing should be implemented. Then, the local improvement method is used to modify the vertices near the boundary surface, which can eliminate most negative Jacobians. The whole mesh quality is improved by minimizing the energy functional (20). Finally, further optimization is implemented to eliminate the Jacobians less than the threshold 0.1.

The mesh quality statistics before and after improvement are listed in Table 3, which shows the significant improvement of the mesh quality. There are thousands of negative Jacobians in the original meshes, while the improved meshes are high quality, either scaled Jacobians or condition numbers are desirable. The cross sections for the three meshes are shown in Figs. 13, 14, 15. It can be seen that the

Table 3 Quality comparison of hexahedral meshes before and after improvement

	Mesh	Mesh size (vertex, quad)	Jacobian		Number of Jacobian				Condition number	
			Worst	Best	Negative	0.0–0.2	0.2–0.6	0.6–1.0	Min	Max
ATcpnα	Original	(196,042, 141,979)	−0.9337	1.0000	45,306	41,747	78,117	970,662	1.0000	4.9e6
	Improved	(299,916, 246,277)	0.0375	1.0000	0	221	190,355	1,779,640	1.0000	328.3
92-grain	Original	(27,720, 25,024)	−0.7993	1.0000	1,783	3,215	14,723	180,473	1.0004	1.2e5
	Improved	(49,072, 44,994)	0.1000	1.0000	0	1,885	47,896	310,171	1.0000	815.6
52-grain	Original	(32,768, 29,791)	−0.6861	1.0000	2,178	4,204	15,157	216,789	1.0000	2.6e15
	Improved	(50,756, 46,695)	0.1002	1.0000	0	226	24,842	348,492	1.0017	1.6e4

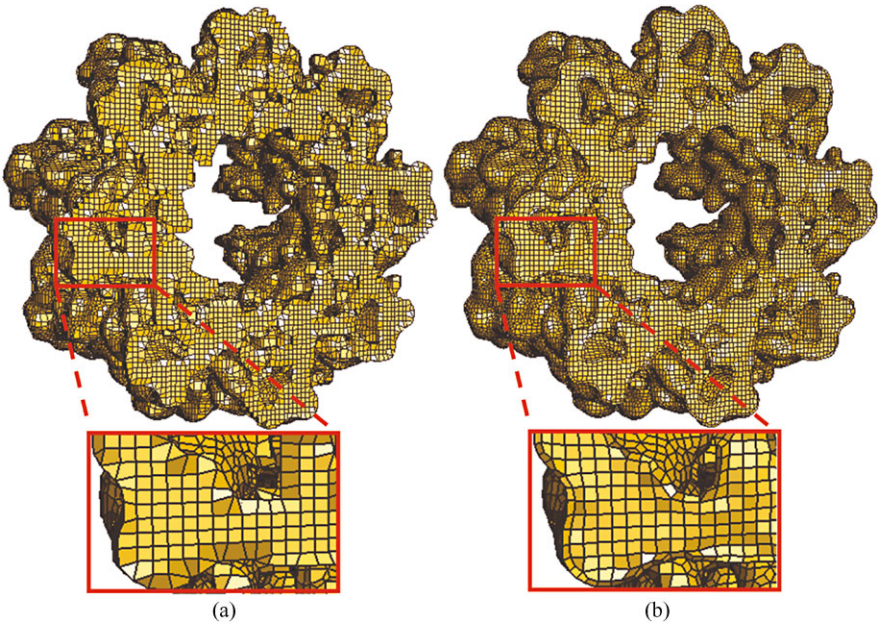


Fig. 13 Cross sections for ATcpnα hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

newly added vertices by the pillowing technique are distributed regularly after mesh improvement.

6 Conclusion

We have developed a series of algorithms to improve the mesh quality of quadrilateral/hexahedral meshes for segmented multiple regions. Our proposed method combines the pillowing technique, geometric flow method, and optimization-based

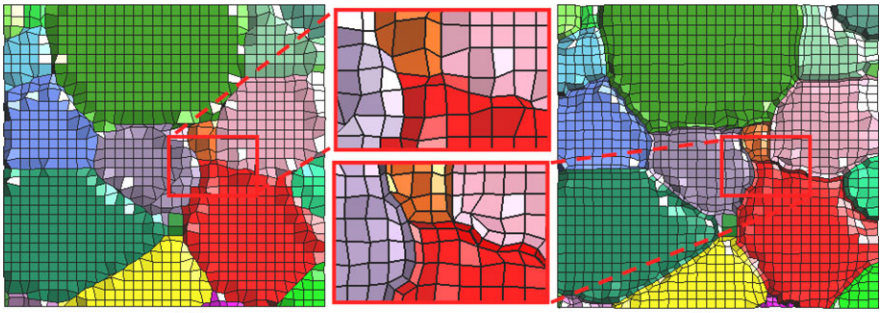


Fig. 14 Cross sections of the 92-grain hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

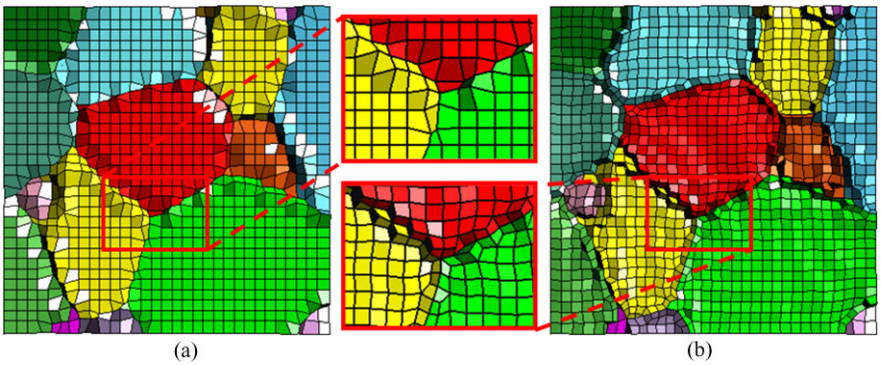


Fig. 15 Cross sections of the 52-grain hexahedral meshes. (**a**) The original mesh; and (**b**) the improved mesh

approaches. The pillowing technique is applied to eliminate the cases that two or more edges/faces of one quadrilateral/hexahedron are located on a boundary curve/surface. Driven by geometric flows, vertices located on boundary curves and boundary surfaces move along the normal direction to remove the zigzag and bumpiness. Energy functionals, which are minimized using L^2-gradient flows, are constructed to regularly distribute vertices and improve vertex Jacobians.

We compared the surface smoothing effects of four typical geometric flows, and utilized the surface diffusion flow, which is feature-preserving and volume-preserving, to smooth surfaces in our quality improvement algorithm. Finally, we validated the proposed method on three application examples. The experimental results and quality statistics results demonstrate the remarkable improvement efficiency of our method. The improved quadrilateral/hexahedral meshes have high quality and the shape feature of boundary curves/surfaces are well preserved.

Acknowledgements The work in this paper was done when J. Leng was a visiting student at the Department of Mechanical Engineering, Carnegie Mellon University. J. Leng, Y. Zhang and

J. Qian were supported in part by Y. Zhang's ONR-YIP award N00014-10-1-0698, an ONR grant N00014-08-1-0653, an AFOSR grant FA9550-11-1-0346, and a NSF/DoD-MRSEC seed grant. J. Leng and G. Xu were supported in part by NSFC key project under the grant 10990013 and Funds for Creative Research Groups of China (grant No. 11021101).

References

1. Bobenko E, Schröder P, Caltech T (2005) Discrete Willmore flow. In: Eurographics symposium on geometry processing, pp 101–110
2. Bryant R (1984) A duality theorem for Willmore surfaces. J Differ Geom 20:23–53
3. Canann A (1991) Plastering and optismoothing: new approaches to automated, 3D hexahedral mesh generation and mesh smoothing. PhD dissertation, Brigham Young University, Provo, UT
4. Desbrun M, Meyer M, Schröder P, Barr AH (1999) Implicit fairing of irregular meshes using diffusion and curvature flow. In: SIGGRAPH'99 proceedings of the 26th annual conference on computer graphics and interactive techniques, Los Angeles, USA, pp 317–324
5. Escher J, Simonett G (1998) The volume preserving mean curvature flow near spheres. Proc Am Math Soc 126(9):2789–2796
6. Field D (1988) Laplacian smoothing and Delaunay triangulation. Commun Appl Numer Methods 4:709–712
7. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Numer Methods Eng 53:1377–1391
8. Freitag L, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Numer Methods Eng 49:109–125
9. Ito Y, Shih A, Soni B (2009) Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. Int J Numer Methods Eng 77:1809–1833
10. Knupp P (2000) Hexahedral mesh untangling and algebraic mesh quality metrics. In: Proceedings of 9th international meshing roundtable, pp 173–183
11. Knupp P (2001) Hexahedral and tetrahedral mesh untangling. Eng Comput 17:261–268
12. Leng J, Zhang Y, Xu G (2011) A novel geometric flow-driven approach for quality improvement of segmented tetrahedral meshes. In: Proceedings of the 20th international meshing roundtable, pp 327–344
13. Li T, McKeag R, Armstrong C (1995) Hexahedral meshing using midpoint subdivision and integer programming. Comput Methods Appl Mech Eng 124:171–193
14. Mitchell S, Tautges T (1995) Pillowing doublets: refining a mesh to ensure that faces share at most one edge. In: Proceedings of the 4th international meshing roundtable, pp 231–240
15. Owen S (1998) A survey of unstructured mesh generation technology. In: Proceedings of the 7th international meshing roundtable, pp 239–267
16. Qian J, Zhang Y, Wang W, Lewis A, Siddiq Qidwai M, Geltmacher A (2010) Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. Int J Numer Methods Eng 82:1406–1423
17. Sapiro G (2001) Geometric partial differential equations and image analysis. Cambridge University Press, Cambridge
18. Schneiders R (1996) A grid-based algorithm for the generation of hexahedral element meshes. Eng Comput 12:168–177
19. Tautges T, Blacker T, Mitchell S (1996) The whisker-weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes. Int J Numer Methods Eng 39:3327–3349
20. Xu G (2008) Geometric partial differential equation methods in computational geometry. Science Press of China, Beijing
21. Zhang Y, Bajaj C (2006) Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Comput Methods Appl Mech Eng 195:942–960

22. Zhang Y, Bajaj C, Sohn B (2005) 3D finite element meshing from imaging data. Comput Methods Appl Mech Eng 194:5083–5106
23. Zhang Y, Bajaj C, Xu G (2005) Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In: Proceedings of the 14th international meshing roundtable, pp 449–468
24. Zhang Y, Hughes T, Bajaj C (2010) An automatic 3D mesh generation method for domains with multiple material. Comput Methods Appl Mech Eng 199:405–415