An Optimization-Based Iterative Approach to Tetrahedral Mesh Smoothing

Zhanheng Gao, Zeyun Yu, and Jun Wang

Abstract The optimal Delaunay triangulation (ODT) is an effective approach in improving the quality of inner vertices of a tetrahedral mesh. Recently it had been extended boundary-optimized Delaunay triangulation (B-ODT), in which both inner and boundary vertices are repositioned by analytically minimizing the $\mathcal{L}^1$ error between a paraboloid function and its piecewise linear interpolation over the neighborhood of each vertex. In the present work, we describe a smoothing method that is based on the B-ODT method but has better performance. We smooth the mesh in an edge-by-edge fashion by adjusting each pair of vertices of every edge. This method has the volume-preserving and sharp-feature-preserving properties. A number of experiments are included to demonstrate the performance of our method.

1 Introduction

The finite element method (FEM) has been a very popular numerical approach for solving partial differential equations (PDEs) in many applications. In the method, the domain over which the PDEs are defined is partitioned into a mesh containing a large number of simple elements, such as triangles and quadrilaterals in 2D cases and tetrahedra and hexahedra in 3D cases [1–6]. The quality of the mesh, typically measured by the minimum and maximum angles, can significantly affect the interpolation accuracy and solution stability of the FEA [7, 8]. Therefore, improving the mesh quality has been an active research area in computational mathematics and computer science. Due to the great popularity in the FEM, 3D tetrahedral meshes will be the focus of our present work.

The methods of mesh quality improvement can be classified into three categories as follows. (1) topology optimization, which modifies the connectivity be-

Z. Gao · Z. Yu (✉) · J. Wang
Department of Computer Science, University of Wisconsin-Milwaukee, 3200 N. Cramer St, Milwaukee, WI 53211, USA
e-mail: yuz@uwm.edu

Z. Gao
College of Computer Science and Technology, Jilin University, 2699 Qianjin St, Changchun, Jilin 130012, China

tween mesh vertices while keeping vertex positions unchanged. The edge- or face-swapping methods are commonly used in topology optimization [9, 10]. (2) vertex insertion/deletion, which inserts/deletes vertices to/from the mesh [10–13]. (3) vertex smoothing, which repositions the coordinates of the vertices while keeping the connectivity unchanged [14–16]. Generally speaking, mesh quality improvement is best achieved when all the three methods are properly combined in the mesh smoothing scheme [10]. In our method described below, we shall focus on the vertex repositioning strategy, i.e. vertex smoothing.

One of the most popular vertex smoothing method is *Laplacian* smoothing, which moves a mesh vertex to the weighted average of its incident vertices [17–19]. If the neighborhood of the vertex is not a convex polyhedron, the Laplacian smoothing may not lead to a well-positioned mesh. Some angle-based methods were proposed for smoothing 2D triangular and 3D surface meshes [20–22]. However, these methods are difficult to extend to 3D tetrahedral meshes. [23] presented a method based on the Centroid Voronoi Tessellation (CVT) concept that is restricted to inner vertices of a mesh. A peeling off operation has to be taken to improve bad tetrahedra on boundaries. [24] proposed a method of smoothing planar quadrilateral meshes. Some researchers presented methods for smoothing hexahedral mesh [25–29]. More recently, some new techniques of vertex smoothing were proposed. [30, 31] presented methods of stretching the vertices of a tetrahedron at one time. The methods were extended by [32] to hexahedral mesh. [33] assigned a quality coordinate for every vertex and calculated the new position by maximizing the combined quality of tetrahedra incident to it. [34] used a metric non-conformity driven method to smooth hybrid meshes such as a mesh with hexahedral and tetrahedral elements.

In addition to the above methods, approaches using numerical optimization to compute the new position of a vertex has been an important branch of the vertex smoothing category. The new position of a vertex is computed by optimizing a function that measures the local or global quality of the mesh [35–44]. In particular, the optimal Delaunay triangulation (ODT) approach [45] tries to minimize the $\mathcal{L}^1$ error between a paraboloid function and its piecewise linear interpolation over the neighborhood of a vertex. This idea has been extended to 3D tetrahedral mesh smoothing in [46]. Despite its great success in mesh quality improvement, the original ODT method was derived to optimize the positions of inner vertices only. In other words, the tetrahedral mesh to be smoothed must possess quality triangles on boundaries. In many real mesh models, however, "bad" tetrahedra often occur near or on the boundaries of a domain [47, 48]. Therefore, in our previous work, we provided an analytical method named boundary-optimized Delaunay triangulation (B-ODT) to find the optimal positions of all mesh vertices, including those on boundaries, by minimizing an $\mathcal{L}^1$ error function that is defined in the incident neighborhood of each vertex. The minimization is an unconstrained quadratic optimization problem and has an exact analytic solution when the coefficient matrix of the problem is positive definite.

In this work, we extend our previous B-ODT method by performing it edge by edge. The new method achieves better results than the original B-ODT method by considering the local configuration of every vertex before performing the B-ODT

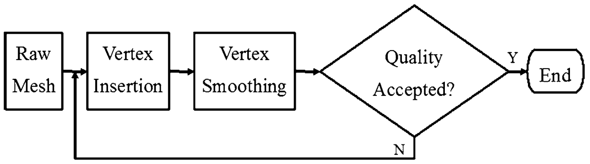


Fig. 1 The framework of our mesh quality improvement method

algorithm. The remainder of the present work is organized as follows. In Sect. 2, we start with a brief introduction to our tetrahedral mesh generation from an initial triangular surface mesh. We then review the ODT and B-ODT methods, provide the edge-based B-ODT (so-called eB-ODT) method. We present some experimental results and quality analysis in Sect. 3, followed by our conclusions in Sect. 4.

2 Methods

The framework of our mesh quality improvement method is shown in Fig. 1. The vertex insertion operation is performed prior to the vertex repositioning. We try to insert as few vertices as possible in order to maintain the size of the original mesh. We give a brief introduction to our tetrahedral mesh generation using an octree-based method in Sect. 2.1. As for vertex smoothing, the algorithms of original B-ODT and eB-ODT are given in Sects. 2.2 and 2.3 respectively.

2.1 Tetrahedral Mesh Generation Algorithm

Our tetrahedral mesh generation algorithm is based on the body centered cubic (BCC) tetrahedral lattice, a common crystal structure in nature with many desirable properties [49]. The BCC lattice is constructed by adding a new node at each cell center and connecting it to the eight vertices of the cell and six neighboring cell centers. The BCC lattice is highly structured and computationally efficient, and has been utilized in various types of numerical simulation. When dealing with a bounded domain, however, the BCC lattice must be carefully remeshed near the domain boundary so that the tetrahedral mesh generated agrees with the given boundary. To this end, our method consists of the following four steps (see Fig. 2 for a two dimensional illustration):

1. Subdivide the octree of an input surface mesh based on Euclidean distance transformation. A few geometric properties of the input mesh are utilized to refine the subdivision adaptively from interior to boundary, and from low curvature to high curvature areas.
2. Compute the sign of every node in the BCC lattice. For each edge of the BCC grid, if the corresponding signs of the two endpoints are different, then calculate the cutting (intersecting) point where the edge crosses the input surface mesh.

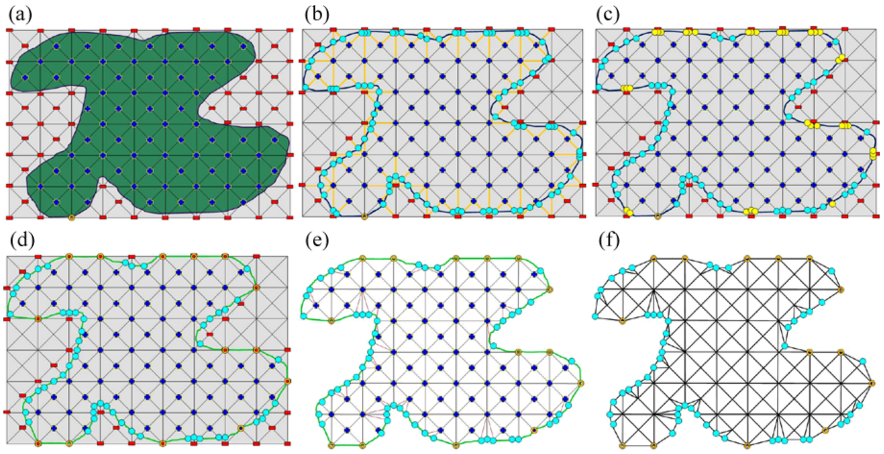


Fig. 2 A two dimensional illustration of our tetrahedral generation algorithm. Note that the octree subdivision is adaptive in our algorithm. However, we do not show the adaptivity here for simplicity. (**a**) Computing the signs for each BCC grid; (**b**) Calculating the cutting points; (**c**) Detecting the "too close" cutting points; (**d**) Snapping the "too close" cutting points to the corresponding BCC lattice grids; (**e**) Decomposing the boundary polyhedra into tetrahedra; (**f**) Obtaining the final tetrahedral mesh

3. Detect the cutting points that are "too close" to the original BCC nodes and snap them to the corresponding nodes. Equivalently, we adjust the sign of that node to zero. We refer to this process as cutting point snapping.
4. Decompose the boundary polyhedra into tetrahedra. For each BCC tetrahedron, if all signs of its vertices are negative (meaning "outside"), we ignore it (we assume that only the interior tetrahedralization is of interest). If all signs are positive (meaning "inside"), we leave it as the final tetrahedron. Otherwise, the tetrahedron is split by the input surface mesh into inside and outside parts and we further decompose the inside part (a polyhedron) into tetrahedra.

2.2 ODT and B-ODT Algorithms

For any vertex $\mathbf{x}_0$ in a tetrahedral mesh $\mathcal{T}$, suppose the neighborhood of $\mathbf{x}_0$ is Ω_0 consisting of a set of tetrahedra $\{\tau\}$. Let $\mathbf{x}_*$ be the smoothing result of $\mathbf{x}_0$ and Ω_* the neighborhood of $\mathbf{x}_*$ (or the union of tetrahedra incident to $\mathbf{x}_*$) in $\mathcal{T}$.

If $\mathbf{x}_0$ is an inner vertex, $\mathbf{x}_*$ can be computed by the following ODT formula [45]:

$$\mathbf{x}_* = \mathbf{x}_0 - \frac{1}{2|\Omega_0|} \sum_{\tau \in \Omega_0} \left(\frac{1}{3} S_\tau \mathbf{n}_\tau \sum_{i=1}^{3} \|\mathbf{x}_{\tau,i} - \mathbf{x}_0\|^2 \right). \tag{1}$$

Here S_τ and $\mathbf{n}_\tau$ are the area and unit normal vector of t_τ, which is the opposite triangle of $\mathbf{x}_0$ in τ, $\mathbf{n}_\tau$ points to the inside of τ, $\mathbf{x}_{\tau,i}$ are the (three) vertices of t_τ.

If $\mathbf{x}_0$ is a boundary vertex, $\mathbf{x}_*$ can be computed by the following B-ODT formula:

$$\mathbf{x}_* = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t} \tag{2}$$

where $\mathbf{s}$ and $\mathbf{t}$ are two orthonormal vectors in the tangent plane of the boundary surface of $\mathcal{T}$ at $\mathbf{x}_0$, the coefficients u and v are computed by solving the following linear equation system:

$$\begin{bmatrix} 2E & G \\ G & 2F \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -H \\ -I \end{bmatrix}. \tag{3}$$

The calculations of E, F, G, H, I are given below in Algorithm 1. Here, we directly give the algorithm of smoothing inner and boundary vertices of $\mathcal{T}$ using ODT and B-ODT methods respectively (Algorithm 1). Note that the tangent plane restriction of $\mathbf{x}_*$ guarantees the volume of the smoothed mesh coincides with that of the original mesh.

Theoretically, both (1) and (2) are the unique solutions of the optimization problem which minimizes the $\mathcal{L}^1$ interpolation error between a paraboloid function $f_I(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_0\|^2$ and its piecewise linear interpolation over Ω_*:

$$Error_* = \|f - f_I\|_{L^1} = \int_{\mathbf{x}\in\Omega_*} \left| f(\mathbf{x}) - f_I(\mathbf{x}) \right| \mathrm{d}\mathbf{x}. \tag{4}$$

Algorithm 1 (ODT and B-ODT smoothing for inner and boundary vertices)

```
for every vertex $\mathbf{x}_0$ do
  a. if $\mathbf{x}_0$ is an inner vertex, then
     $\mathbf{x}_*$ is computed by (1).
  b. else
     $\mathbf{x}_*$ is computed using the following scheme:
      i. Compute the normal vector of the tangent plane at $\mathbf{x}_0$, then select
         two orthogonal unit vectors $\mathbf{s}$, $\mathbf{t}$ on the tangent plane.
     ii. Compute the following coefficients:
         A. $E = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{s}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
         B. $F = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{t}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
         C. $G = -\frac{1}{60}\sum_{i=1}^{m}[\mathbf{s}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1}) + \mathbf{t}(\mathbf{Y}_i + \mathbf{Y}_{i+1}) \times \mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})]$
         D. $H = \frac{1}{12}\mathbf{s}\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1}) \times \mathbf{s}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
         E. $I = \frac{1}{12}\mathbf{t}\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1})\mathbf{t}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
    iii. Solve the linear system (3).
     iv. Compute $\mathbf{x}_*$ using $\mathbf{x}_* = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t}$.
```

Here, $\mathbf{Y}_i = \mathbf{y}_i - \mathbf{x}_0$, $\{\mathbf{y}_i\}_{i=1}^{m}$ are the neighboring vertices of $\mathbf{x}_0$ on the boundary of the tetrahedral mesh $\mathcal{T}$. The order of $\mathbf{y}_i$ is determined in the following way: for any $i = 1, \dots, m$, the cross product between $\overrightarrow{\mathbf{x}_0\mathbf{y}_i}$ and $\overrightarrow{\mathbf{x}_0\mathbf{y}_{i+1}}$ points to the outside of Ω_0 (let $\mathbf{y}_{m+1} = \mathbf{y}_1$), $L_\tau = \sum_{j=1}^{3} \|\mathbf{x}_{\tau,j} - \mathbf{x}_0\|^2$.

For a boundary vertex $\mathbf{x}_0$, in order to preserve the sharp features, we further restrict $\mathbf{x}_*$ moving along the features of the mesh. Here, we refer to the feature direction at $\mathbf{x}_0$ as the line that passes through $\mathbf{x}_0$ and has the minimal curvature value among all the directions. This line is on the tangent plane, thus the volume is still preserved when $\mathbf{x}_*$ moves along this feature line. The direction of the feature line is found by computing the eigenvalues of the following tensor voting matrix at $\mathbf{x}_0$:

$$M = \sum_{i=1}^{m} S_i \mathbf{n}_i \mathbf{n}_i^T. \tag{5}$$

Here S_i is the area of surface triangle $\Delta\mathbf{x}_0\mathbf{y}_i\mathbf{y}_{i+1}$ and $\mathbf{n}_i = (n_{ix}, n_{iy}, n_{iz})^T$ is the unit normal vector of $\Delta\mathbf{x}_0\mathbf{y}_i\mathbf{y}_{i+1}$. The matrix M is a positive definite matrix and has three orthogonal eigenvectors. The feature line is determined in the following way. Suppose that the three eigenvalues of M are μ_0, μ_1, μ_2 with $\mu_0 \geq \mu_1 \geq \mu_2$ and $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$ are the corresponding eigenvectors. If $\mu_0 \gg \mu_1 \approx \mu_2 \approx 0$, then the neighborhood of $\mathbf{x}_0$ corresponds to a planar feature. In this case, the above Algorithm 1 is used to smooth $\mathbf{x}_0$. If $\mu_0 \approx \mu_1 \gg \mu_2 \approx 0$, then $\mathbf{x}_0$ lies on an crease (linear) feature and the direction of the crease is $\mathbf{e}_2$. In this case, the following Algorithm 2 is used to smooth $\mathbf{x}_0$. If $\mu_0 \approx \mu_1 \approx \mu_2 \gg 0$, then $\mathbf{x}_0$ is at a corner which should not be changed during the vertex smoothing process.

Algorithm 2 (B-ODT smoothing with feature preserving)

```
for every crease vertex $\mathbf{x}_0$ do
    a. Set the feature direction at $\mathbf{x}_0$ to be $\mathbf{d} = \mathbf{e}_2/\|\mathbf{e}_2\|$.
    b. Compute the following coefficients:
        i. $A = \frac{1}{4}|\Omega_0| - \frac{1}{60}\sum_{i=1}^{m} \mathbf{d}(\mathbf{Y}_i + \mathbf{Y}_{i+1})\mathbf{d}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
        ii. $B = \frac{1}{12}\mathbf{d}(\sum_{\tau\in\Omega_*} S_\tau \mathbf{n}_\tau L_\tau) - \frac{1}{60}\sum_{i=1}^{m}(\mathbf{Y}_i^2 + \mathbf{Y}_{i+1}^2 + \mathbf{Y}_i\mathbf{Y}_{i+1})\mathbf{d}(\mathbf{Y}_i \times \mathbf{Y}_{i+1})$
    c. Compute $\mathbf{x}_*$ as $\mathbf{x}_* = \mathbf{x}_0 + f\mathbf{d}$ with $f = -\frac{B}{2A}$.
```

2.3 Edge-Based B-ODT Algorithm

In practice, the improvement of $\mathbf{x}_0$ is always affected by the configuration of the vertices around $\mathbf{x}_0$. When the vertices around $\mathbf{x}_0$ has good configuration, the quality can be significantly improved. Based on this observation, we presented a modified strategy here to smooth a tetrahedral mesh: smoothing the mesh in an edge-by-edge way. That is, to smooth the two end vertices of each edge recursively. By this way, the angle quality can be improved more than simply performing Algorithm 2. The detail is given in the following algorithm:

Algorithm 3 (eB-ODT smoothing for boundary vertices)

```
for every edge $\mathbf{e}$ in the mesh (Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be the two vertices of $\mathbf{e}$), do
    a. Smooth $\mathbf{x}_0$ using Algorithm 1 or Algorithm 2 according to the type of $\mathbf{x}_0$
    b. Smooth $\mathbf{x}_1$ using Algorithm 1 or Algorithm 2 according to the type of $\mathbf{x}_1$
    c. Compute $s$, which is the sum of the movements of $\mathbf{x}_0$ and $\mathbf{x}_1$
    d. If $s \leq \varepsilon$, go to next edge
       else, go to step a
```

3 Results

The proposed eB-ODT algorithms were tested on several tetrahedral meshes generated from triangular surface meshes that serve as the boundaries of the domains. For every mesh, the smoothing process shown in Fig. 1 is repeated for 20 times. The mesh smoothing results are summarized in Table 1. The comparisons between the eB-ODT algorithm (Algorithm 3) and several other approaches, including the ODT algorithm, B-ODT algorithm, topology optimization and the Natural ODT algorithm [46], are also provided in Table 1. In Figs. 3–9, the original and smoothed meshes are compared and from the histograms we can see significant improvement of dihedral angles in these meshes.

We compare the smoothing results by using the ODT, B-ODT and eB-ODT algorithms. In Table 1, all the minimum and maximum dihedral angles by using the B-ODT algorithm are better than those by the ODT algorithm and the results by using eB-ODT are better than B-ODT, especially on the Retinal model. Note that the minimum dihedral angle in Retinal model is very small and likely occurs on the boundary of the model. Therefore, the B-ODT and eB-ODT algorithm can perform much better than the original ODT method.

Although the topology optimization is utilized in many mesh smoothing algorithms, this technique alone may not always improve the quality of a mesh. To show this, we smooth all the meshes in Table 1 using only the topology optimization and compare the results with those obtained by using our eB-ODT algorithm. From Table 1, we can see that the ability of improving mesh quality by using topology optimization alone is limited, compared to the eB-ODT algorithm.

The tetrahedral mesh in Fig. 3 is generated by tetrahedralizing randomly-sampled point set on a unit sphere [50]. There are 642 points on the sphere and 87 inner vertices are inserted by the tetrahedralization algorithm. The minimum and maximum

Table 1 Comparisons of dihedral angles using different methods

Model	Algorithm	Vertex number	min	max
Random Sphere	Original mesh	729	5.86°	164.70°
	eB-ODT	767	18.20°	145.56°
	B-ODT	731	15.20°	150.25°
	ODT	729	6.28°	162.46°
	Topology optimization	729	5.86°	164.70°
	NODT	729	6.21°	173.64°
2Cmp	Original mesh	10415	5.57°	163.24°
	eB-ODT	10488	21.68°	145.56°
	B-ODT	10415	18.10°	147.21°
	ODT	10415	11.64°	158.06°
	Topology optimization	10415	5.57°	163.24°
	NODT	10415	10.70°	157.19°
Retinal	Original mesh	14921	1.25°	173.85°
	eB-ODT	15030	19.86°	160.14°
	B-ODT	14948	15.10°	164.58°
	ODT	14921	1.29°	168.13°
	Topology optimization	14921	1.25°	172.09°
	NODT	14921	0.00°	179.99°
RyR	Original mesh	18585	6.19°	170.74°
	eB-ODT	18601	22.57°	143.56°
	B-ODT	18585	18.52°	149.25°
	ODT	18585	10.34°	158.32°
	Topology optimization	18585	6.19°	170.74°
	NODT	18585	7.78°	162.74°
2Torus	Original mesh	4635	5.96°	164.92°
	eB-ODT	4731	21.37°	146.81°
	B-ODT	4656	16.92°	152.05°
	ODT	4635	9.46°	157.53°
	Topology optimization	4635	6.85°	164.75°
	NODT	4635	0.01°	179.98°
FanDisk	Original mesh	9131	6.04°	164.98°
	eB-ODT	9173	20.32°	154.96°
	B-ODT	9162	16.80°	160.53°
	ODT	9131	9.59°	163.53°
	Topology optimization	9131	6.78°	164.98°
	NODT	9131	0.08°	179.86°

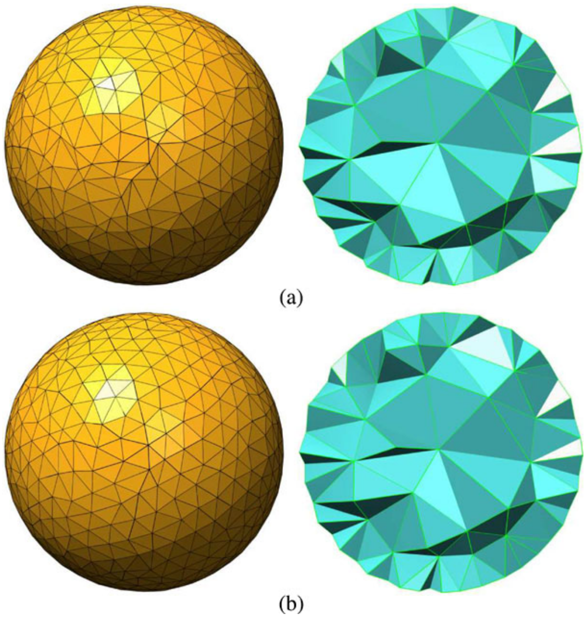


Fig. 3 The original mesh model (**a**) and the smoothed result (**b**). In both meshes, the outer and cross-section views are shown. The minimum dihedral angles of these two meshes are 5.86° and 18.20° respectively, and the maximum dihedral angles are 164.70° and 145.56° respectively

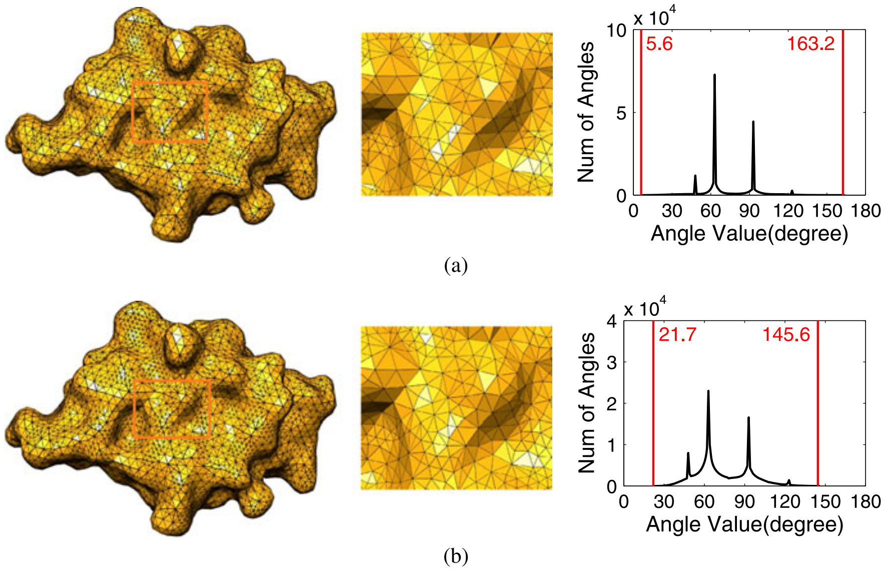
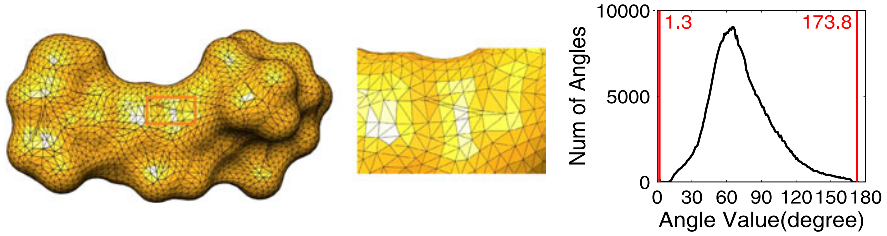
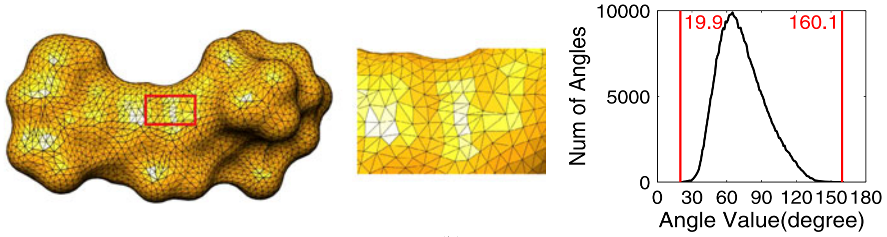


Fig. 4 Original and smoothed 2CMP models. The minimum dihedral angles of these two meshes are 5.57° and 21.68° respectively, and the maximum dihedral angles are 163.24° and 145.56° respectively

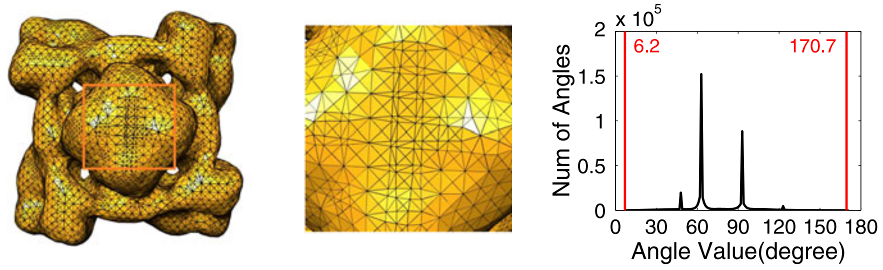


(a)

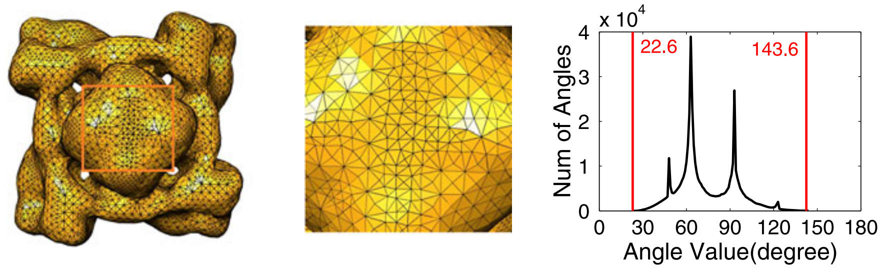


(b)

Fig. 5 Original and smoothed Retinal models. The minimum dihedral angles of these two meshes are 1.25° and 19.86° respectively, and the maximum dihedral angles are 173.85° and 160.14° respectively



(a)



(b)

Fig. 6 Original and smoothed RyR models. The minimum dihedral angles of these two meshes are 6.19° and 22.57° respectively, and the maximum dihedral angles are 170.74° and 143.56° respectively

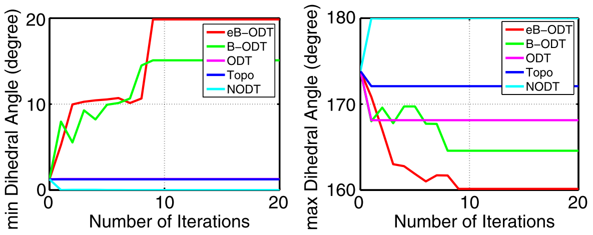


Fig. 7 The convergence of minimum and maximum dihedral angles with respect to the number of iterations on the Retinal model using the eB-ODT algorithm. Note that on the *left* the curves of ODT and topology optimization are almost identical

dihedral angles of this Random Sphere model are 5.86° and 164.70° respectively. After 20 times of running the eB-ODT algorithm, the minimum and maximum dihedral angles are improved to 18.20° and 145.56° respectively. Note that the distribution of the boundary vertices of the smoothed mesh is much more uniform than that of the original mesh, demonstrating that the eB-ODT algorithm can smooth both inner and boundary vertices in a tetrahedral mesh.

The eB-ODT algorithm is also tested on tetrahedral meshes generated from several biomedical molecules: 2CMP molecule in Fig. 4, Retinal molecule in Fig. 5 and Ryanodine receptor (RyR) in Fig. 6. The quality of 2CMP and RyR meshes reaches the best after no more than 10 iterations although all the models in Table 1 are processed 20 times. In Fig. 7, we demonstrate the convergence of minimum and maximum dihedral angles with respect to the number of iterations on the Retinal model using the eB-ODT algorithm.

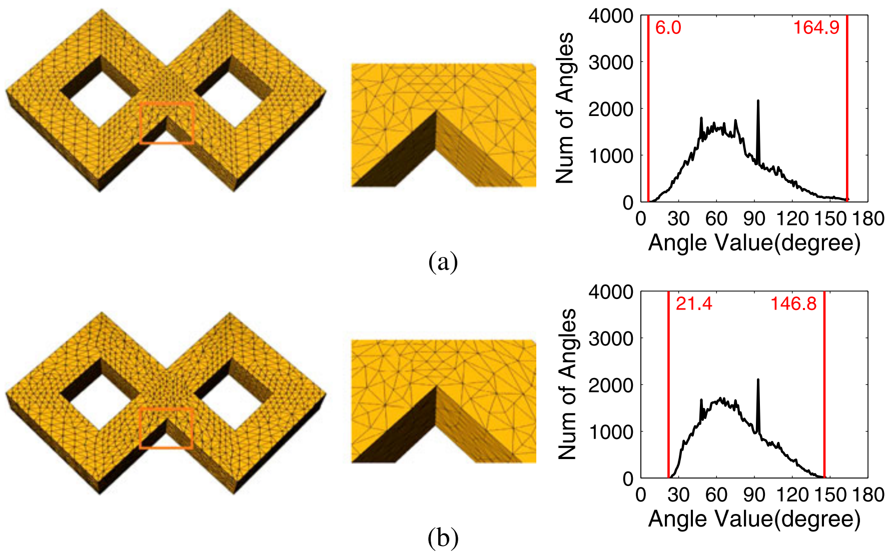
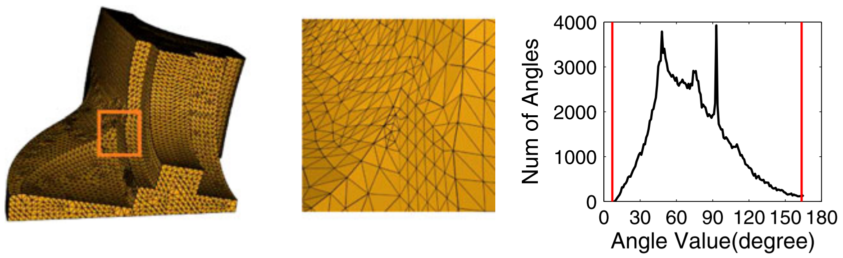
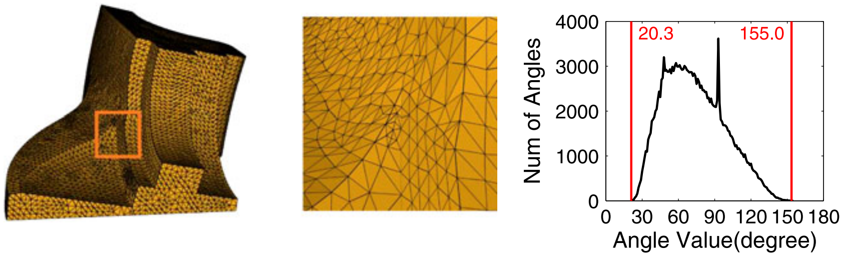


Fig. 8 Original and smoothed 2Torus models. The minimum dihedral angles of these two meshes are 5.96° and 21.37° respectively, and the maximum dihedral angles are 164.92° and 146.81° respectively



(a)



(b)

Fig. 9 Original and smoothed FanDisk models. The minimum dihedral angles of these two meshes are 6.04° and 20.32° respectively, and the maximum dihedral angles are 164.98° and 154.96° respectively

Table 2 Relative Hausdorff distance between original and smoothed meshes

Models	Random Sphere	2Cmp	Retinal	RyR	2Torus	FanDisk
Rel. Hausdorff distance	0.40%	1.11%	0.52%	0.77%	0.18%	0.25%

The 2Torus (Fig. 8) and FanDisk (Fig. 9) models show the feature-preserving property of the eB-ODT algorithm. In order to measure the difference between the original and smoothed meshes, we compute the relative Hausdorff distances between the surface meshes of the original and smoothed models, as shown in Table 2. Here, the Hausdorff distance is first computed using the standard definition and then scaled as follows. Let h be the absolute Hausdorff distance between the original and smoothed meshes, and L be the largest side length of the bounding box of the original mesh. The relative Hausdorff distances is defined by $\frac{h}{L}$, which measures the difference of the original and smoothed models relative to the size of the original model. From Table 2 we can see that the relative Hausdorff distances between the original and smoothed models are very small showing that our eB-ODT algorithm preserves the shape of the original models quite well.

The original ODT has also been extended by [46] to 3D tetrahedral mesh smoothing and the method is called Natural ODT (NODT). The NODT method computes the new position of a boundary vertex $\mathbf{x}_0$ in a tetrahedral mesh $\mathcal{T}$ by adding a certain amount of compensation to the weighted centroid of the neighborhood of $\mathbf{x}_0$. The

Table 3 Comparison of running time (20 iterations)

	Random Sphere	2Cmp	Retinal	RyR	2Torus	FanDisk
eB-ODT	305.84 s	5175.6 s	10272.8 s	9086.0 s	2258.4 s	4522.0 s
NODT	11.85 s	159.80 s	323.12 s	291.24 s	64.08 s	124.24 s

compensation is a weighted sum of the normal vectors of the boundary triangles around $\mathbf{x}_0$. Although boundary vertices are considered in the NODT method, the new positions calculated have to be projected onto the boundary of $\mathcal{T}$ to preserve the volume and shape of the original mesh. Therefore, the NODT method does not optimize the positions for boundary vertices. The smoothing results by using the afore-mentioned NODT method are shown in Table 1, where we can see that our eB-ODT algorithm significantly outperforms the NODT method. Sometimes the results obtained by the NODT method are even worse than the original meshes. The running time of eB-ODT and NODT is compared in Table 3.

4 Conclusions

We described a method of simultaneously smoothing both inner and boundary vertices of a tetrahedral mesh under a unified optimization framework. The eB-ODT algorithm presented can preserve sharp features very well and is guaranteed to preserve the volume of the original mesh. For every boundary vertex, the optimal position is computed by solving a linear system. The algorithm is numerically robust and easy to implement because the order of the linear equation system is only degree 2. The experimental results have shown the effectiveness of the proposed method.

Acknowledgements The work described was supported in part by an NIH Award (Number R15HL103497) from the National Heart, Lung, and Blood Institute (NHLBI) and by a subcontract from the National Biomedical Computation Resource (NIH Award Number P41 RR08605). The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors.

References

1. Djidjev H (2000) Force-directed methods for smoothing unstructured triangular and tetrahedral meshes. In: 9th international meshing roundtable, pp 395–406
2. Phillippe P, Baker T (2001) A comparison of triangle quality measures. In: 10th international meshing roundtable, pp 327–340
3. Ohtake Y, Belyaev A, Bogaevski I (2001) Mesh regularization and adaptive smoothing. Comput Aided Des 33(11):789–800
4. Knupp PM (2002) Algebraic mesh quality metrics. SIAM J Sci Comput 23(1):193–218
5. Knupp PM (2003) Algebraic mesh quality metrics for unstructured initial meshes. Finite Elem Anal Des 39(3):217–241

6. Brewer M, Freitag Diachin L, Knupp P, Leurent T, Melander D (2003) The mesquite mesh quality improvement toolkit. In: 12th international meshing roundtable, pp 239–250
7. Babuska I, Aziz AK (1976) On the angle condition in the finite element method. SIAM J Numer Anal 13(2):214–226
8. Shewchuk JR (2002) What is a good linear element? Interpolation, conditioning, and quality measures. In: 11th international meshing roundtable, pp 115–126
9. Freitag LA, Ollivier-Gooch C (1997) Tetrahedral mesh improvement using swapping and smoothing. Int J Numer Methods Eng 40(21):3979–4002
10. Klingner BM, Shewchuk JR (2008) Aggressive tetrahedral mesh improvement. In: 16th international meshing roundtable. Springer, Berlin, pp 3–23
11. Chew LP (1997) Guaranteed-quality Delaunay meshing in 3D. In: 13th annual symposium on computational geometry, pp 391–393
12. Nave D, Chrisochoides N, Chew LP (2004) Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains. Comput Geom, Theory Appl 28(2–3):191–215
13. Escobar JM, Montenegro R, Montero G, Rodríguez E, Gonzáez-Yuste JM (2005) Smoothing and local refinement techniques for improving tetrahedral mesh quality. Comput Struct 83(28–30):2423–2430
14. Bank RE, Smith RK (1997) Mesh smoothing using a posteriori error estimates. SIAM J Numer Anal 34(3):979–997
15. Freitag LA (1997) On combining Laplacian and optimization-based mesh smoothing techniques. In: Trends in unstructured mesh generation. AMD, vol 220, pp 37–43
16. Canann SA, Tristano JR, Staten ML (1998) An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes. In: 7th international meshing roundtable, pp 419–494
17. Herrmann LR (1976) Laplacian-isoparametric grid generation scheme. J Eng Mech Div 102(5):749–907
18. Field DA (1988) Laplacian smoothing and Delaunay triangulations. Commun Appl Numer Methods 4(6):709–712
19. Hansbo P (1995) Generalized Laplacian smoothing of unstructured grids. Commun Numer Methods Eng 11(5):455–464
20. Zhou T, Shimada K (2000) An angle-based approach to two-dimensional mesh smoothing. In: 9th international meshing roundtable, pp 373–384
21. Xu H, Newman TS (2006) An angle-based optimization approach for 2D finite element mesh smoothing. Finite Elem Anal Des 42(13):1150–1164
22. Yu Z, Holst MJ, McCammon JA (2008) High-fidelity geometric modeling for biomedical applications. Finite Elem Anal Des 44(11):715–723
23. Du Q, Wang D (2003) Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. Int J Numer Methods Eng 56:1355–1373
24. Freitag L, Plassmann P (2001) Local optimization-based untangling algorithms for quadrilateral meshes. In: 10th international meshing roundtable, pp 397–406
25. Li X, Freitag LA, Freitag LA (1999) Optimization-based quadrilateral and hexahedral mesh untangling and smoothing techniques. Technical report, Argonne National Laboratory
26. Knupp PM (2001) Hexahedral and tetrahedral mesh untangling. Eng Comput 17(3):261–268
27. Knupp PM (2000) Hexahedral mesh untangling & algebraic mesh quality metrics. In: 9th international meshing roundtable, pp 173–183
28. Delanaye M, Hirsch C, Kovalev K (2003) Untangling and optimization of unstructured hexahedral meshes. Comput Math Math Phys 43(6):807–814
29. Menédez-Díaz A, González-Nicieza C, Álvarez-Vigil AE (2005) Hexahedral mesh smoothing using a direct method. Comput Geosci 31(4):453–463
30. Vartziotis D, Athanasiadis T, Goudas I, Wipper J (2008) Mesh smoothing using the geometric element transformation method. Comput Methods Appl Mech Eng 197(45–48):3760–3767
31. Vartziotis D, Wipper J, Schwald B (2009) The geometric element transformation method for tetrahedral mesh smoothing. Comput Methods Appl Mech Eng 199(1–4):169–182

32. Vartziotis D, Wipper J (2011) A dual element based geometric element transformation method for all-hexahedral mesh smoothing. Comput Methods Appl Mech Eng 200(9–12):1186–1203
33. Xu K, Cheng Z-Q, Wang Y, Xiong Y, Zhang H (2009) Quality encoding for tetrahedral mesh optimization. Comput Graph 33(3):250–261. IEEE international conference on shape modelling and applications 2009
34. Sirois Y, Dompierre J, Vallet M-G, Guibault F (2010) Hybrid mesh smoothing based on Riemannian metric non-conformity minimization. Finite Elem Anal Des 46(1–2):47–60
35. Parthasarathy V, Kodiyalam S (1991) A constrained optimization approach to finite element mesh smoothing. Finite Elem Anal Des 9(4):309–320
36. Canann SA, Stephenson MB, Blacker T (1993) Optismoothing: an optimization-driven approach to mesh smoothing. Finite Elem Anal Des 13(2–3):185–190
37. Chen CL, Szema KY, Chakravarthy SR (1995) Optimization of unstructured grid. In: 33rd aerospace sciences meeting and exhibit, Reno, NV, January, pp 1–10. AIAA 95-0217
38. Zavattieri PD, Dari EA, Buscaglia GC (1996) Optimization strategies in unstructured mesh generation. Int J Numer Methods Eng 39(12):2055–2071
39. Freitag Diachin L, Knupp P (1999) Tetrahedral element shape optimization via the Jacobian determinant and condition number. In: 8th international meshing roundtable, pp 247–258
40. Knupp PM (2000) Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I—a framework for surface mesh optimization. Int J Numer Methods Eng 48(3):401–420
41. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Numer Methods Eng 49(1–2):109–125
42. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Numer Methods Eng 53(6):1377–1391
43. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Simultaneous untangling and smoothing of tetrahedral meshes. Comput Methods Appl Mech Eng 192(25):2775–2787
44. Mezentsev A (2004) A generalized graph-theoretic mesh optimization model. In: 13th international meshing roundtable, pp 255–264
45. Chen L, Xu J (2004) Optimal Delaunay triangulations. J Comput Math 22(2):299–308
46. Tournois J, Wormser C, Alliez P, Desbrun M (2009) Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. ACM Trans Graph 28:75–1759
47. Labelle F, Shewchuk JR (2007) Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Trans Graph 26:57–15710
48. Zhang Y, Hughes TJR, Bajaj CL (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415
49. Molino N, Bridson R, Teram J, Fedkiw R (2003) A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: 12th international meshing roundtable, pp 103–114
50. Si H, Gärtner K, Fuhrmann J (2010) Boundary conforming Delaunay mesh generation. Comput Math Math Phys 50(1):38–53