Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 435648, 9 pages
http://dx.doi.org/10.1155/2015/435648

*Research Article*

# Smoothing Algorithm for Planar and Surface Mesh Based on Element Geometric Deformation

**Shuli Sun, Minglei Zhang, and Zhihong Gou**

*LTCS, Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China*

Correspondence should be addressed to Shuli Sun; sunsl@mech.pku.edu.cn

Received 18 September 2014; Accepted 18 November 2014

Academic Editor: Chenfeng Li



Smoothing is one of the basic procedures for improvement of mesh quality. In this paper, a novel and efficient smoothing approach for planar and surface mesh based on element geometric deformation is developed. The presented approach involves two main stages. The first stage is geometric deformation of all the individual elements through a specially designed two-step stretching-shrinking operation (SSO), which is performed by moving the vertices of each element according to a certain rule in order to get better shape of the element. The second stage is to determine the position of each node of the mesh by a weighted average strategy according to quality changes of its adjacent elements. The suggested SSO-based smoothing algorithm works efficiently for triangular mesh and can be naturally expanded to quadrilateral mesh, arbitrary polygonal mesh, and mixed mesh. Combined with quadratic error metric (QEM), this approach may be also applied to improve the quality of surface mesh. The proposed method is simple to program and inherently very suitable for parallelization, especially on graphic processing unit (GPU). Results of numerical experiments demonstrate the effectiveness and potential of this method.

## 1. Introduction

As an elementary technique, mesh quality improvement has been widely used in many applications, such as mesh generation [1–5], mesh simplification [6–8], dynamic grid or mesh deformation [9–14], and other mesh processing procedures. Due to its importance, mesh quality improvement technique has received a lot of attention and great progress has already been made in recent years [15–34]. Mesh improvement approach can be basically divided into two main categories, topological optimization and smoothing (also called geometrical optimization). Topological optimization changes the topology of a mesh, that is, the node-element connectivity relationship, while smoothing or geometrical optimization improves mesh quality by simply moving or adjusting node positions without changing the topology of mesh. This paper will focus on the latter, smoothing.

Theoretically the optimization-based smoothing procedure [16, 21, 22, 29] would lead to better results. However, the capacity of improvements is limited when applying such kind of computationally expensive procedure to practical problem. In many works, for example, in [18], smoothing was commonly performed by Laplacian smoothing technique, that is, simply shifting each interior node (free vertex) to the centroid of the surrounding polygon or polyhedron. Although it is computationally inexpensive and easy to implement, however, it may produce occasionally invalid or illegal elements that are unacceptable in numerical analysis. Chen and Xu [25] presented the concept of optimal Delaunay triangulation and developed a quality smoothing scheme for triangular mesh [26] by minimizing the interpolation error among all triangulations with the same number of vertices in the local patch. This optimization-based smoothing method [26–28] could solve the corresponding optimization problem explicitly and thus the computational cost is as low as that of Laplacian smoothing. However, this method is only suitable for triangular mesh and cannot be expanded to other types of mesh.

Recently Vartziotis et al. [33, 34] developed a novel smoothing method named GETMe (geometric element transformation method) for triangular and quadrilateral mesh based on element geometric transformation. This method first improves element shape by two-step geometric transformation and then updates positions of moved nodes to achieve the purpose of smoothing. It is not necessary to construct or solve optimization problem, so high efficiency of the method is expected. Test results show that the method

achieves excellent performance in effect of smoothing and algorithm efficiency. Unfortunately, it is unsuccessful to expand this approach to polygonal elements with more than four edges (e.g., pentagon and hexagon).

Inspired by the work of Vartziotis et al. [33, 34], this paper proposes a new and efficient smoothing approach for planar and surface mesh based on element geometric deformation. The approach includes two main stages. In the first stage, for each individual element, all vertices are moved by a purposely designed two-step stretching-shrinking operation (SSO) in order to get better shape of the element. The second stage is to determine the position of each node of the mesh by a weighted average strategy according to quality changes of its adjacent elements. The proposed SSO-based smoothing approach has the universal scheme and can be conveniently expanded from triangular mesh to quadrilateral mesh, arbitrary polygonal mesh, and mixed mesh. Combined with QEM, this approach may be also applied to improve the quality of surface mesh. Furthermore, the suggested method is simple to program and inherently very suitable for parallelization, especially on GPU.

In the rest of this paper we first review the idea and basic procedures of GETMe [33, 34] and then the two-step SSO and the corresponding smoothing approach for planar mesh are presented. Next the smoothing approach for surface mesh is proposed by introducing QEM. Finally, some numerical experiments are given to demonstrate the effectiveness and feasibility of the method.

## 2. Related Work

Recently a novel smoothing approach named GETMe (geometric element transformation method) for triangular and quadrilateral mesh was proposed by Vartziotis et al. [33, 34]. This approach first improves element shape by two-step geometric transformation and then updates positions of moved nodes to achieve the purpose of smoothing. It is not necessary to construct or solve optimization problem, so high efficiency of the method is expected.

The method first operates on a counterclockwise oriented triangle with pairwise disjoint vertices $z_i^{(0)}$, $i \in \{0, 1, 2\}$. Taking the edge $z_0^{(0)} z_1^{(0)}$ of the triangle, for example, rotating it by an angle $\theta$ clockwise with $z_0^{(0)}$ as the center, then a new vertex $z_1^{(1/2)}$ defined as the intersection of two subsidiary lines (the perpendicular of $z_0^{(0)} z_1^{(0)}$ in $z_1^{(0)}$ and the edge $z_0^{(0)} z_1^{(0)}$ rotated clockwise by angle $\theta$ with $z_0^{(0)}$ as the center) is obtained by

$$\begin{aligned} z_1^{(1/2)} &= z_1^{(0)} - w\left(z_1^{(0)} - z_0^{(0)}\right) \\ &= w z_0^{(0)} + (1 - w)\, z_1^{(0)}, \end{aligned} \tag{1}$$

where $w = i \tan\theta$ and complex numbers are used to simplify notation.

Applying the same procedure to the other two edges, a clockwise rotated new triangle with the vertices $z_0^{(1/2)}$, $z_1^{(1/2)}$, and $z_2^{(1/2)}$ is obtained. To avoid the rotational effect, a similar counterclockwise transformation with angle $\theta$ by

$$\begin{aligned} z_i^{(1)} &= z_i^{(1/2)} - w\left(z_{(i+1) \bmod 3}^{(1/2)} - z_i^{(1/2)}\right) \\ &= (1 + w)\, z_i^{(1/2)} - w z_{(i+1) \bmod 3}^{(1/2)}, \quad i = 0, 1, 2, \end{aligned} \tag{2}$$

is applied to get a new triangle with the vertices $z_0^{(1)}$, $z_1^{(1)}$, and $z_2^{(1)}$.

It is proved that such geometric element transformations preserve the element centroid, and an arbitrary triangle will gradually converge to an equilateral triangle after enough successive transformations [33].

The above transformations usually enlarge the size of the original triangle. In the event that the element does not have any boundary nodes, scaling is applied so that the new element preserves its original perimeter. If an edge of the element belongs to a boundary, the element is scaled according to the length of this edge and is moved so that the nodes belonging to this edge return to their original positions [33].

This geometric transformation can be generalized leading to an additional degree of freedom in controlling the speed in which an element becomes regular [34]. For quadrilateral elements, similar transformation can be also conducted [34]. After successively performing such transformation, an arbitrary quadrilateral will become a regular square.

In [33], the authors suggested a sequential GETMe approach based on successively improving elements with the lowest quality to complete smoothing of the mesh. Initially, the poor quality elements are rated. During the sequential process, the transformation algorithm is applied to the "worst" element, until the mesh quality reaches the desired level or termination criteria are met. After each transformation, all the affected elements are updated. Alternatively, a simultaneous GETMe approach [34] was presented for smoothing mixed planar meshes. It is based on transforming all elements simultaneously. Updated node positions are obtained as weighted means of the transformed element nodes.

Testing shows that the method achieves excellent performance in the effect of smoothing and algorithm efficiency. Unfortunately, it is unsuccessful to expand this approach to polygonal elements with more than four edges (e.g., pentagon and hexagon).

## 3. Smoothing Algorithm for Planar Mesh

A new and universal smoothing approach for planar triangular mesh based on element geometric deformation is first proposed in this section. The presented approach includes two main stages. The first stage is geometric deformation of all the individual triangles. For each triangle, the vertices are moved by a specially designed two-step stretching-shrinking operation (SSO) in order to get better shape of the element. The second stage is to determine the position of each node of the mesh by a weighted average strategy according to the quality changes of its adjacent triangles. Next, the geometric deformation operation for triangle is first proposed and then is followed by its natural extensions to quadrilateral and arbitrary polygon.

*3.1. Geometric Deformation of Single Triangles.* As for the algorithm in the first stage, geometric deformation of each individual triangle can be performed by two operations, stretching and shrinking. The stretching operation will enlarge the size of the original triangle but change the element towards regular shape. In order to preserve its original size, a scaling or shrinking operation must be applied to the enlarged triangle with better shape. The detail of SSO is given as follows.

For a counterclockwise oriented triangle with pairwise disjoint vertices $p_i^{(0)} = [x_i^{(0)}, y_i^{(0)}]^T$, $i \in \{0, 1, 2\}$, the stretching operation is performed by pulling each vertex $p_i^{(0)}$ out along the perpendicular direction $n_i$ of its opposite edge (see Figure 1).

Then we get an enlarged new triangle with the vertices $p_i^{(1')}$ calculated by

$$p_i^{(1')} = p_i^{(0)} + \lambda n_i, \quad i = 0, 1, 2, \tag{3}$$

where $\lambda$ can be intuitively understood as a stretching factor to control the degree of geometric stretching. Large value of $\lambda$ means more significant improvement for element shape or quality. There are many choices to define $\lambda$. Here, the following expression in terms of the average length $l$ of the original element and the quality measurement $q$ ($q \in [0, 1]$, $q$ takes the value of 1 for regular element and 0 for degenerated element) is adopted to reflect the fact that only small stretching is needed when the element approaches to regular shape ($q$ gets close to 1):

$$\lambda = (1 - q)\, l. \tag{4}$$

In (4), the quality measurement $q$ for triangle can be taken as $2r/R$, where $r$ and $R$ represent the radius of incircle and circumcircle of the triangle, respectively.

After preforming the stretching operation, the shape or quality of the original poor triangle can be significantly improved. This procedure, however, will usually enlarge the size of the original triangle. In order to preserve its original size, a scaling or shrinking operation shown in Figure 2 must be applied to the enlarged triangle with better shape.

The shrinking operation does not change the shape of the element. In general, there are two ways for shrinking, keeping perimeter or area of the element unchanged. In practice, test results show that there is no big difference between these two ways. In this paper, for the reason of computational cost, we keep the perimeter (the average length) of the element unchanged before and after deformation. In addition, the shrinking operation needs to keep the position of centroid unchanged for each element. Thus, we obtain the shrinking triangle with new vertices of

$$p_i^{(1)} = c^{(0)} + k\left(p_i^{(1')} - c^{(1')}\right), \quad i = 0, 1, 2, \tag{5}$$

where $c^{(0)} = (\sum_{i=0}^{2} p_i^{(0)})/3$ and $c^{(1')} = (\sum_{i=0}^{2} p_i^{(1)})/3$ represent the centroid of the triangle before and after stretching, respectively. The shrinking factor $k$ is defined as the ratio of the average length of the element before and after stretching.

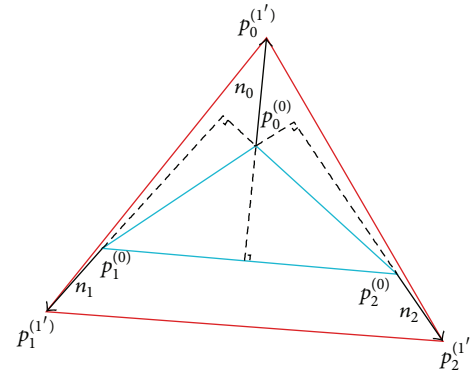



Figure 1: Geometric deformation of a triangle (stretching operation).

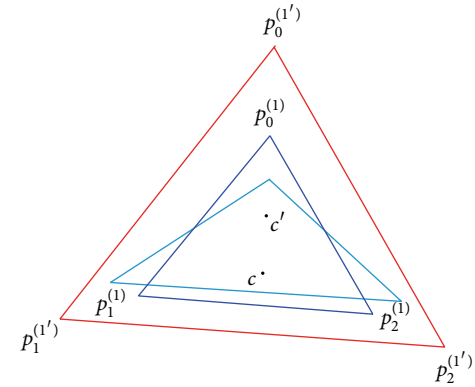



Figure 2: Geometric deformation of a triangle (shrinking operation).

Stretching and shrinking operations make up a complete geometric element deformation process. The final shape of the triangle is only determined by the stretching factor $\lambda$ and the shrinking factor $k$ together, which can be easily calculated by coordinates of the vertices.

Figure 3 shows the geometric deformation process of a triangle. A poor shape triangle gradually becomes regular after 5 successive cycles of SSO.

The proposed SSO is numerically efficient and is more simple to implement compared with the GETMe method. More important, the approach has the universal scheme and can be conveniently expanded to quadrilateral elements and arbitrary polygonal elements.

*3.2. Geometric Deformation of Single Quadrilaterals.* The SSO for quadrilaterals is a natural extension of the above procedure. For a counterclockwise oriented quadrilateral with four vertices $p_i^{(0)}$, $i \in \{0, 1, 2, 3\}$, shown in Figure 4, stretching operation is performed by pulling each vertex $p_i^{(0)}$ out along the perpendicular direction $n_i$ of the diagonal which links its two adjacent vertices. This operation will usually enlarge the size of the original element; in the meantime, the shape or quality of the original poor element can be significantly improved. Similarly, a shrinking operation is also applied to the enlarged element with better shape to preserve its original size.

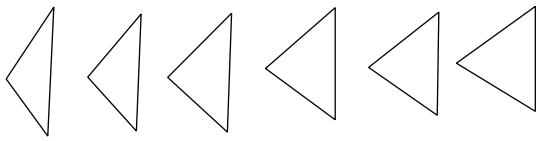


Figure 3: Geometric deformation process of a triangle.

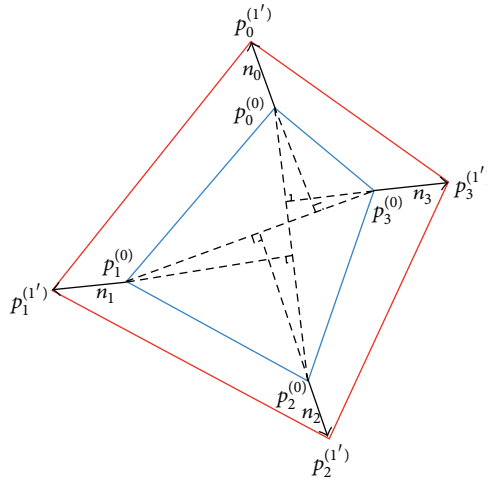



Figure 4: Geometric deformation of a quadrilateral (stretching operation).

The complete SSO is formulated by

$$p_i^{(1')} = p_i^{(0)} + \lambda n_i,$$
$$p_i^{(1)} = c^{(0)} + k\left(p_i^{(1')} - c^{(1')}\right), \quad i = 0, 1, 2, 3, \tag{6}$$

where $\lambda, k$ denote the stretching and shrinking factor and $c^{(0)}$ and $c^{(1')}$ represent the centroid of the element before and after deformation, respectively. Also the stretching factor $\lambda$ is used to control the degree of geometric stretching. For quadrilateral elements, the stretching factor $\lambda$ takes the same expression in (4) but replacing $q$ by a quality measurement for quadrilaterals.

Figure 5 shows the geometric deformation process of a quadrilateral. A poor shape quadrilateral gradually becomes regular after 5 successive cycles of SSO.

*3.3. Geometric Deformation of Arbitrary Polygonal Elements.* In accordance with the deformation of quadrilateral elements, the extension of SSO to arbitrary polygonal elements is quite straightforward. For instance, the sketch of stretching deformation for a pentagon is illustrated in Figure 6. One can easily derive the deformation formulation similar to (6).

For such kind of mesh smoothing approach discussed in the following subsection, each element is deformed individually in the first stage, and the only thing we need to do is to perform SSO for each element independently without caring about the connection with its neighbor elements. Therefore, the proposed algorithm has the inherent advantage of dealing with mixed mesh.

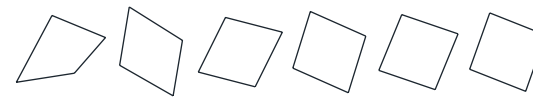


Figure 5: Geometric deformation process of a quadrilateral.

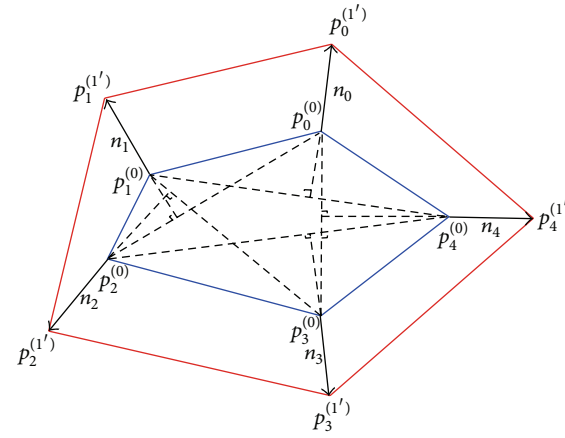



Figure 6: Geometric deformation of a pentagon (stretching operation).

*3.4. SSO-Based Smoothing Algorithm.* This subsection gives the algorithm to complete smoothing approach based on SSO. The weighted average strategy is selected in the second stage to determine the position of each node of the mesh according to the quality changes of its adjacent elements. This strategy is aimed at improving the average quality of the whole mesh, while the quality of bad elements can be also improved. Before updating the positions of nodes, each element needs to conduct SSO independently in the first stage to get the temporary vertices (the vertices of the deformed element with better shape). Then the position of node $p_i$ can be updated by

$$p_i = \frac{\sum_e (\Delta q_{i,e})\, p_{i,e}^T}{\sum_e (\Delta q_{i,e})}, \tag{7}$$

where $p_{i,e}^T$ and $\Delta q_{i,e}$ denote the corresponding temporary vertex and quality change of the adjacent element $e$, respectively. According to (7), the element with larger change in quality accounts for larger weight in determining the final position of the node, which leads to more emphasis on the transformed nodes of low quality elements.

The main procedure of the SSO-based smoothing algorithm for planar mesh is described as follows.

*Algorithm 1* (SSO-based smoothing algorithm for planar mesh).

(1) For each node of the mesh, find all of its adjacent elements.

(2) For each element, conduct geometric deformation independently by SSO and save the coordinates of all temporary vertices.

(3) For each node of the mesh, calculate the weighted average of the corresponding temporary vertices by (7) to get the position of this node.

(4) If the mesh quality reaches the desired level or termination criteria are met, stop; otherwise, return to step (2) for next cycle.

The proposed smoothing procedure has a fast convergence speed; usually 10~20 cycles will achieve quite good result. In addition, the main calculations, including SSO for elements and position updating for nodes with the time complexity of $O(n)$, have very good parallelizability inherently. Therefore, it is suitable for developing parallel scheme of the method, especially on GPU. Actually, a GPU implementation is completed by the authors. Some test results given in Section 5 show remarkable speedup gained by GPU parallel technology.

## 4. Smoothing Algorithm for Surface Mesh

The SSO-based smoothing algorithm can be also applied to surface mesh. Different from planar mesh, we must consider an important issue; that is, the movement of the node is not allowed to deviate from the original surface too much. In this section, we introduce the quadratic error metric (QEM) [6] which has been widely used in surface simplification to control the moving distance of the nodes away from the original surface in the process of geometric deformation.

*4.1. The Basic Idea of QEM [6].* It is easy to observe that, in original surface mesh, each node $p_i$ is the solution of the intersection of a set of planes, $S(p_i)$. Then the QEM of node $p_i$, $\Delta(p_i)$, is defined as the sum of squared distances to its planes:

$$K_p = \begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix}, \qquad Q_i = \sum_{p \in S(p_i)} K_p, \tag{8}$$

$$\Delta(p_i) = p_i^T Q_i p_i.$$

In (8) $a, b, c, d$ represent the coefficients of the plane defined by the equation $ax + by + cz + d = 0$ with $a^2 + b^2 + c^2 = 1$; $p_i = [x_i, y_i, z_i, 1]^T$ denotes the augmented coordinate vector of node $p_i$.

The QEM can be used to describe the deviation degree of a node away from the original surface. Note that the initial value of QEM for each node is 0, since each node in original mesh lies in the planes of all its incident elements. In smoothing of surface mesh, we need to control the value of QEM to ensure that the element geometric deformation is acceptable.

*4.2. SSO-QEM Based Smoothing for Surface Mesh.* Before performing element geometric deformation to surface mesh, the quadratic error matrix $Q_i$ in (8) for each node must be calculated in advance. For each element, construct the plane according to the coordinates of its vertices and find the coefficients of $a, b, c, d$, and then calculate the matrix $K_p$ for each element according to (8). By using the relationship between nodes and faces of the mesh, for each node, find all of its adjacent elements, and then add the matrix $K_p$ of its adjacent elements up to get the error matrix $Q_i$. With the aid of the quadratic error matrix $Q_i$, we can easily calculate the error of the node $p_i$ by (8) when the node is moved during smoothing process. Please note that the quadratic error matrix $Q_i$ only needs to be calculated once.

The SSO for element geometric deformation needs to be revised to meet the requirement of surface mesh. For simplicity, we take triangular surface mesh as an example. Figure 7 is a local region of a triangular surface mesh, and the element $p_0^{(0)} p_1^{(0)} p_2^{(0)}$ will be deformed by the geometric transformation. The deformation is restrained in the original plane during the stretching and shrinking operations; namely, the position of each vertex only changes in the original plane of the element.

The stretching operation is exactly the same as that in planar case. The stretching factor $\lambda$ is taken as the same expression in (4). The three vertices of the element, respectively, move along the direction normal to their opposite edge. The new positions of $p_i^{(1')}$ can be calculated by (3).

In the scaling or shrinking process, the basic principle remains to keep the centroid of the element unchanged as that in planar case, but the shrinking factor $k$ is no longer the same in (5). Representing the three vertices after shrinking with $p_0^{(1)}$, $p_1^{(1)}$, and $p_2^{(1)}$, the total quadratic error due to element deformation can be formulated as

$$\Delta_e = \sum_{i=0}^{2} {p_i^{(1)}}^T Q_i p_i^{(1)}. \tag{9}$$

Recalling the SSO formulation in (3) and (5), one can see that the total quadratic error $\Delta_e$ is only decided by the stretching factor $\lambda$ and shrinking factor $k$; that is,

$$\Delta_e = E(\lambda, k). \tag{10}$$

Since the stretching factor $\lambda$ is given in terms of the average length $l$ of the original element and the quality measurement $q$, the total quadratic error $\Delta_e$ of the element is only the function of shrinking factor $k$. The remaining thing we need to do is to find the optimal value of $k$ by minimizing the total quadratic error $\Delta_e$, which can be easily solved by analytical or numerical manner. With the optimal value of $k$ obtained, the new positions of $p_i^{(1)}$ can be calculated by (5).

The main procedure of so-called SSO-QEM based smoothing algorithm for surface mesh is summarized as follows.

*Algorithm 2* (SSO-QEM based smoothing algorithm for surface mesh).

(1) For each element, formulate the equation of the plane according to the coordinates of its vertices, and calculate the matrix $K_p$ according to (8).

(2) For each node of the mesh, find all of its adjacent elements; then add the matrix $K_p$ of its adjacent elements up to get the error matrix $Q_i$.

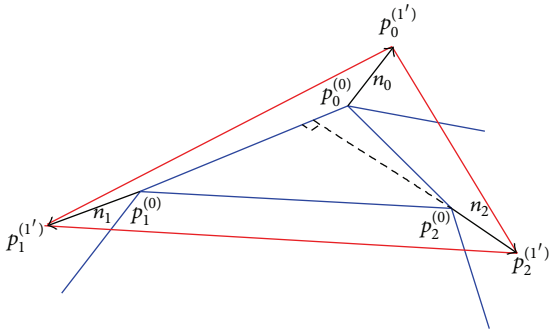



Figure 7: Element stretching for triangular surface mesh.

(3) For each element, project the coordinates of its vertices to the plane, and then conduct geometric deformation independently by stretching operation; find the optimal value of the shrinking factor $k$ by minimizing the total quadratic error $\Delta_e$; conduct shrinking operation independently and save the coordinates of all temporary vertices.

(4) For each node of the mesh, calculate the weighted average of the corresponding temporary vertices by (7) to get the position of this node.

(5) If the mesh quality reaches the desired level or termination criteria are met, stop; otherwise, return to step (3) for next cycle.

With the aid of QEM to control the moving distance of the nodes away from the original surface, the proposed SSO-QEM based smoothing algorithm is able to improve the quality of surface mesh while maintaining the shape of surface as much as possible.

## 5. Experimental Results

Several examples are selected to demonstrate the effectiveness and efficiency of the proposed approach. Both examples are tested on Windows 7 system with Intel (R) Xeon (R) CPU (2.67 GHz) and 24 GB memory.

First a triangular mesh with 7229 nodes and 13942 elements is tested (see Figure 8). The quality of initial mesh is very poor. Several smoothing methods, including Laplacian method, GETMe method, and the proposed SSO-based method, are conducted to improve the quality of mesh. The experimental results are listed in Table 1. From the results, it can be seen that the proposed SSO-based method has slight advantages over GETMe method in average quality and gives much better result in quality of the worst element. In this test, Laplacian method even gives negative value of the quality, which means invalid or illegal elements are produced.

The second example is a planar quadrilateral mesh shown in Figure 9, with the boundary nodes fixed. The mesh contains 1089 nodes and 1024 elements, and the initial quality is also very poor. The experimental results are listed in Table 2. In this test the performances of the proposed method and GETMe method are almost the same. Both of them give better results in quality of the worst element compared with Laplacian method.

Table 1: Quality statistics of triangular mesh by different methods.

| Methods | Average quality | Quality of the worst element |
|---|---|---|
| Initial mesh | 0.77 | 0.002 |
| Laplacian | 0.95 | −0.860 |
| GETMe | 0.94 | 0.004 |
| SSO | 0.96 | 0.023 |

Table 2: Quality statistics of quadrilateral mesh by different methods.

| Methods | Average quality | Quality of the worst element |
|---|---|---|
| Initial mesh | 0.37 | 0.03 |
| Laplacian | 0.74 | 0.08 |
| GETMe | 0.73 | 0.11 |
| SSO | 0.74 | 0.10 |

Table 3: Smoothing results of surface meshes by the proposed method.

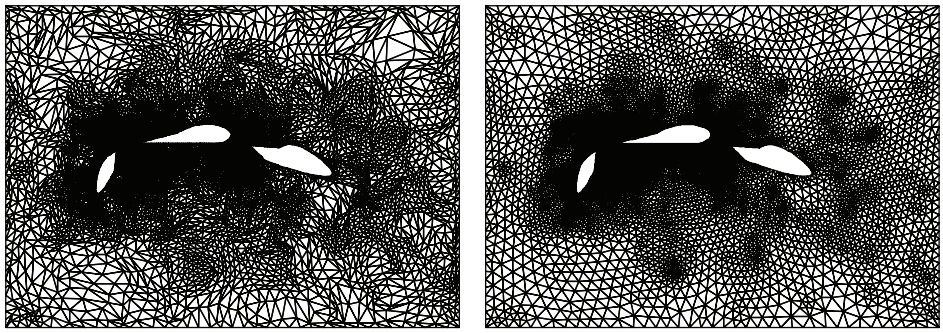
| Mesh | Initial quality | Quality after smoothing | Running time (s) | Geometric error |
|---|---|---|---|---|
| Dragon | 0.58 | 0.74 | 3.66 | 0.8696 |
| Bunny | 0.63 | 0.77 | 1.41 | 0.4022 |

In the following two surface meshes of dragon and bunny [35] (Figure 10) are tested by the proposed SSO-QEM based smoothing approach. The dragon mesh consists of 50000 nodes and 19994 triangular surface elements, while the bunny mesh contains 34835 nodes and 7996 triangular surface elements. Table 3 lists the smoothing results. It can be seen that the proposed SSO-QEM algorithm has obviously improved the quality of each mesh while the surface is well maintained. Here, the geometric error in Table 3, defined as the ratio of Hausdorff distance [7] and the average size of the mesh, is adopted to measure the difference between the new and original surface meshes.

Finally, an example for testing the performance of GPU implementation of the proposed method is given. The code is realized on the platform CUDA 4.0. The type of GPU used for this test is Tesla C2050 (1.15 GHz) with 3034 MB memory. The testing planar mesh consists of 445182 nodes and 887622 triangles. In order to compare the efficiency, 50 cycles or iterations are run on CPU and GPU, respectively (actually it does not need that much of iterations to reach convergence). Table 4 gives the comparison of smoothing result and running time. More than 300 times speedup is gained by GPU implementation which displays great superiority and potential of the proposed method in the aspect of computational efficiency.

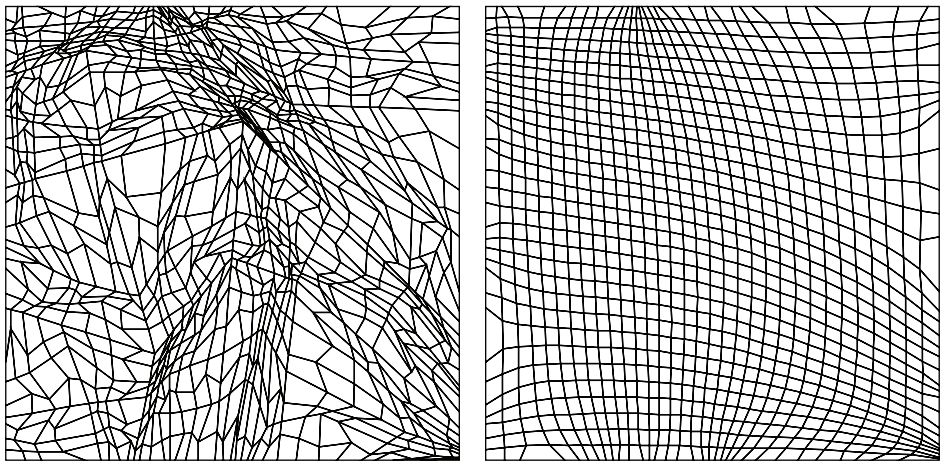
## 6. Conclusions

This paper presents a novel and efficient smoothing algorithm for 2D planar and surface mesh based on element geometric deformation. The main features of the proposed method are summarized as follows.



(a) Initial mesh (b) Smoothed mesh by suggested method

Figure 8: A triangular mesh before and after smoothing.



(a) Initial mesh (b) Smoothed mesh by suggested method

Figure 9: A quadrilateral mesh before and after smoothing.

Table 4: Comparison of smoothing result and efficiency on CPU and GPU.

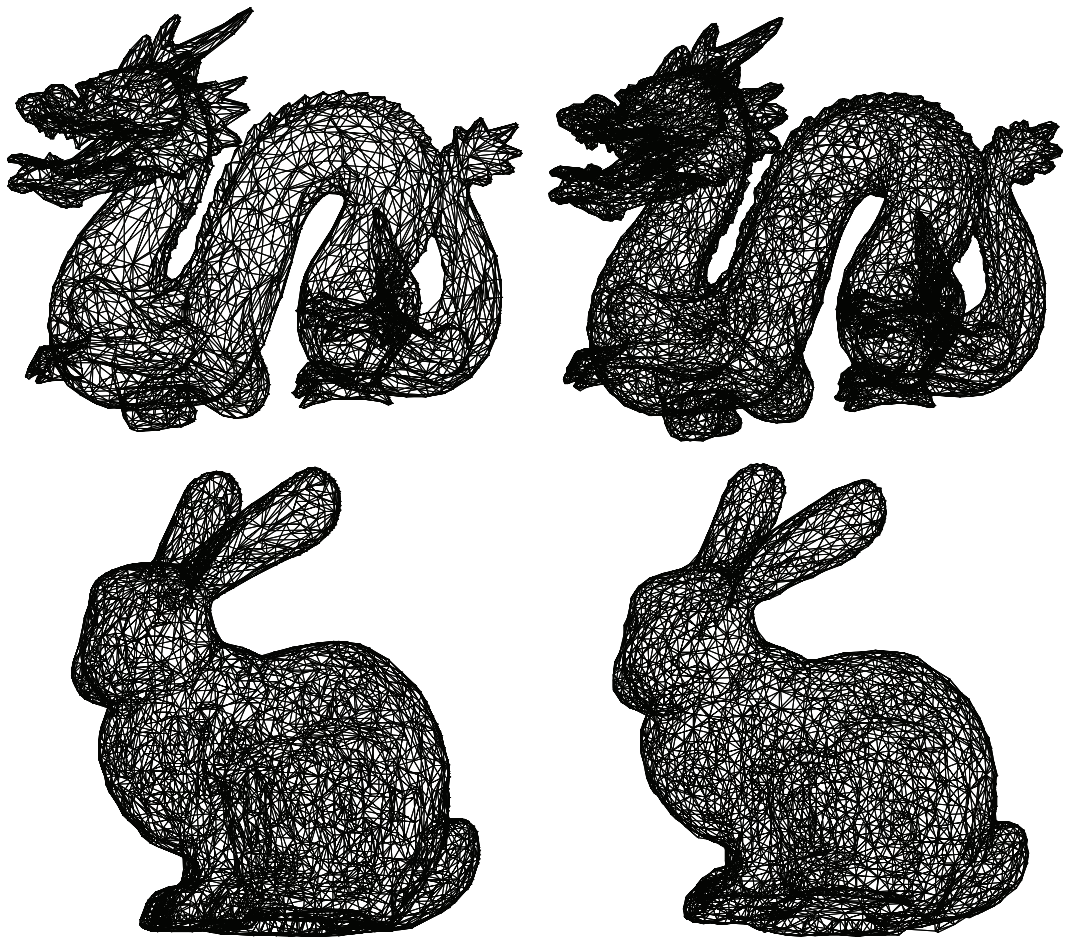
| Methods | Average quality | Quality of the worst element | Running time (s) |
|---|---|---|---|
| Initial | 0.876 | 0.246 | — |
| Smoothing on CPU | 0.952 | 0.504 | 276.05 |
| Smoothing on GPU | 0.955 | 0.504 | 0.8150 |

(1) The smoothing of mesh involves two main stages, namely, geometric deformation of all the elements and updating positions of all the nodes. These two stages can be performed independently.

(2) The element geometric deformation consists of two operations: stretching and shrinking. The stretching operation changes the shape and improves the quality of the element, while the shrinking operation preserves the centroid and size of the element. These operations mainly involve two parameters, the stretching factor and shrinking factor. As for the algorithm in this paper, the stretching factor is decided by the size and quality of the element, while the shrinking factor for planar mesh is defined as the ratio of the average length of the element before and after stretching. Single elements will converge to regular shape after successively geometric deformations.

(3) In the stage of node updating, the final position of each node is calculated by taking weighted average of the corresponding temporary nodes obtained from element stretching and shrinking operations. The element with larger change in quality accounts for larger weight in determining the final position of the node, which leads to more emphasis on the transformed nodes of low quality elements.

(4) The proposed SSO-based smoothing approach has the universal scheme and can be conveniently expanded from triangular mesh to quadrilateral mesh, arbitrary polygonal mesh, and mixed mesh. Moreover, each element is deformed individually in the geometric deformation stage without caring about the connection with its neighbor elements; therefore, the proposed algorithm has the inherent advantage of dealing with mixed mesh.



(a) Initial mesh (b) Smoothed mesh by suggested method

Figure 10: Surface mesh before and after smoothing.

(5) The main computation efforts, including SSO for elements and position updating for nodes, have very good parallelizability inherently. It is very suitable for developing parallel scheme of the proposed method, especially on GPU. Some test results show remarkable speedup gained by GPU parallel technology, which is very important for dealing with large scale meshes.

(6) For smoothing surface mesh, quadratic error metric (QEM) is introduced to element geometric deformation to control the moving distance of the nodes away from the original surface. The SSO for element geometric deformation only needs slight revision by shifting the shrinking factor, which can be solved by minimizing the total quadratic error due to element deformation.

To conclude, the proposed method has the universal scheme and very simple formulation. It works efficiently for smoothing triangular mesh, quadrilateral mesh, arbitrary polygonal mesh, and mixed mesh. It can be also applied to improve quality of surface mesh by incorporating with QEM. Moreover, it is very suitable for developing parallel implementation. Results of numerical experiments demonstrate the effectiveness and potential of this method. Further study will focus on extensions of the method to mixed volume meshes.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors would like to thank the support of National Basic Research Program of China (2010CB731503), the National Natural Science Foundation of China (11172004, 10772004), and Beijing Municipal Natural Science Foundation (1102020).

## References

[1] J. Liu, "Automatic triangulation of n-D domains," in *Proceedings of the CAD/Graphics '91*, pp. 238–241, Hangzhou, China, 1991.

[2] E. A. Dari and G. C. Buscaglia, "Mesh optimization: how to obtain good unstructured 3D finite element meshes with not-so-good mesh generators," *Structural Optimization*, vol. 8, no. 2-3, pp. 181–188, 1994.

[3] P. L. George, "Improvements on Delaunay-based three-dimensional automatic mesh generator," *Finite Elements in Analysis and Design*, vol. 25, no. 3-4, pp. 297–317, 1997.

[4] Q. Du and D. Wang, "Recent progress in robust and quality Delaunay mesh generation," *Journal of Computational and Applied Mathematics*, vol. 195, no. 1-2, pp. 8–23, 2006.

[5] H. Si, "Constrained Delaunay tetrahedral mesh generation and refinement," *Finite Elements in Analysis and Design*, vol. 46, no. 1-2, pp. 33–46, 2010.

[6] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the Conference on Computer Graphics (SIGGRAPH '97)*, pp. 209–216, Los Angeles, Calif, USA, August 1997.

[7] J. Li and G. Lu, "Mesh simplification and optimization with edge collapse and mass-spring model," *Journal of Computer-Aided Design and Computer Graphics*, vol. 18, no. 3, pp. 426–432, 2006.

[8] S. Lv, M. Zhang, and S. Sun, "Topological optimization for triangular mesh based on simplification and subdivision," *Journal of Computer-Aided Design & Computer Graphics*, vol. 26, no. 8, pp. 1225–1231, 2014 (Chinese).

[9] N. Qin, X. Liu, and H. Xia, "An efficient moving grid algorithm for large deformation," *Modern Physics Letters B*, vol. 19, no. 28-29, pp. 1499–1502, 2005.

[10] X. Liu, N. Qin, and H. Xia, "Fast dynamic grid deformation based on Delaunay graph mapping," *Journal of Computational Physics*, vol. 211, no. 2, pp. 405–423, 2006.

[11] S. Sun, B. Chen, J. Liu, and M. W. Yuan, "An efficient implementation scheme for the moving grid method based on Delaunay graph mapping," in *Proceedings of the 2nd International Symposium on Computational Mechanics and the 12th International Conference on the Enhancement and Promotion of Computational Methods in Engineering and Science, ISCM II and EPMESC XII*, vol. 1233, pp. 1046–1051, Hong Kong, 2009.

[12] S. Sun and B. Chen, "An algebraic deformation approach for moving grid based on barycentric coordinates," in *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE '10)*, pp. 1–4, Wuhan, China, December 2010.

[13] S. Sun and M. Yuan, "High quality mesh deformation method for large scale unstructured hybrid grid," *IACM Expressions*, vol. 29, pp. 9–12, 2011.

[14] S. Sun, J. Liu, B. Chen, and M. Yuan, "Advances in quality improvement and deformation for unstructured mesh," in *Proceedings of the Symposium on the Development of Computational Mechanics and Computational Methods in Engineering and Science*, Macau, China, April 2011.

[15] B. Joe, "Construction of three-dimensional improved-quality triangulations using local transformations," *SIAM Journal on Scientific Computing*, vol. 16, no. 6, pp. 1292–1307, 1995.

[16] P. D. Zavattieri, E. A. Dari, and G. C. Buscaglia, "Optimization strategies in unstructured mesh generation," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 12, pp. 2055–2071, 1996.

[17] L. A. Freitag and C. Ollivier-Gooch, "Tetrahedral mesh improvement using swapping and smoothing," *International Journal for Numerical Methods in Engineering*, vol. 40, no. 21, pp. 3979–4002, 1997.

[18] S. H. Lo, "Optimization of tetrahedral meshes based on element shape measures," *Computers and Structures*, vol. 63, no. 5, pp. 951–961, 1997.

[19] L. A. Freitag and C. Ollivier-Gooch, "A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency," *International Journal of Computational Geometry & Applications*, vol. 10, no. 4, pp. 361–382, 2000.

[20] L. A. Freitag and P. Plassmann, "Local optimization-based simplicial mesh untangling and improvement," *International Journal for Numerical Methods in Engineering*, vol. 49, no. 1-2, pp. 109–125, 2000.

[21] S. Sun and J. Liu, "An efficient optimization procedure for tetrahedral meshes by chaos search algorithm," *Journal of Computer Science and Technology*, vol. 18, no. 6, pp. 796–803, 2003.

[22] Z. J. Chen, J. R. Tristano, and W. Kwok, "Construction of an objective function for optimization-based smoothing," *Engineering with Computers*, vol. 20, no. 3, pp. 184–192, 2004.

[23] J. Liu, S. Sun, and D. Wang, "Optimal tetrahedralization for small polyhedron: a new local transformation strategy for 3-D mesh generation and mesh improvement," *Computer Modeling in Engineering and Sciences*, vol. 14, no. 1, pp. 31–43, 2006.

[24] J. Liu and S. Sun, "Small polyhedron reconnection: a new way to eliminate poorly-shaped tetrahedra," in *Proceedings of the 15th International Meshing Roundtable*, pp. 241–258, Sandia National Laboratories, 2006.

[25] L. Chen and J.-c. Xu, "Optimal delaunay triangulations," *Journal of Computational Mathematics*, vol. 22, no. 2, pp. 299–308, 2004.

[26] L. Chen, "Mesh smoothing schemes based on optimal Delaunay triangulations," in *Proceedings of the 13th International Meshing Roundtable*, pp. 109–120, Sandia National Laboratories, 2004.

[27] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun, "Variational tetrahedral meshing," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 617–625, 2005.

[28] S. Sun, H. Bao, M. Liu, and Y. Yuan, "Quality improvement algorithm for tetrahedral mesh based on optimal delaunay triangulation," *Intelligent Information Management*, vol. 5, no. 6, pp. 191–195, 2013.

[29] L. F. Diachin, P. Knupp, T. Munson, and S. Shontz, "A comparison of two optimization methods for mesh quality improvement," *Engineering with Computers*, vol. 22, no. 2, pp. 61–74, 2006.

[30] R. V. Garimella, M. J. Shashkov, and P. M. Knupp, "Triangular and quadrilateral surface mesh quality optimization using local parametrization," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 9–11, pp. 913–928, 2004.

[31] I. Semenova, N. Kozhekin, V. Savchenko, and I. Hagiwara, "A general framework for analysis and comparison of surface mesh optimization techniques," *Engineering with Computers*, vol. 21, no. 2, pp. 91–100, 2005.

[32] J. M. Escobar, G. Montero, R. Montenegro, and E. Rodriguez, "An algebraic method for smoothing surface triangulations on a local parametric space," *International Journal for Numerical Methods in Engineering*, vol. 66, no. 4, pp. 740–760, 2006.

[33] D. Vartziotis, T. Athanasiadis, I. Goudas, and J. Wipper, "Mesh smoothing using the geometric element transformation method," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 3760–3767, 2008.

[34] D. Vartziotis and J. Wipper, "The geometric element transformation method for mixed mesh smoothing," *Engineering with Computers*, vol. 25, no. 3, pp. 287–301, 2009.

[35] "The Stanford 3D Scanning Repository," https://graphics.stanford.edu/data/3Dscanrep/.