AN ANGLE-BASED APPROACH TO TWO-DIMENSIONAL MESH SMOOTHING

Tian Zhou[1] and Kenji Shimada[2*]

[1]Carnegie Mellon University, Pittsburgh, PA, U.S.A. tzhou@andrew.cmu.edu
[2]Carnegie Mellon University, Pittsburgh, PA, U.S.A. shimada@cmu.edu

ABSTRACT

We present an effective and easy-to-implement angle-based smoothing scheme for triangular, quadrilateral and tri-quad mixed meshes. For each mesh node our algorithm compares all the pairs of adjacent angles incident to the node and adjusts these angles so that they become equal in the case of a triangular mesh and a quadrilateral mesh, or they form the ideal ratio in the case of a tri-quad mixed mesh. The size and shape quality of the mesh after this smoothing algorithm is much better than that after Laplacian smoothing. The proposed method is superior to Laplacian smoothing by reducing the risk of generating inverted elements and increasing the uniformity of element sizes. The computational cost of our smoothing method is yet much lower than optimization-based smoothing. To prove the effectiveness of this algorithm, we compared errors in approximating a given analytical surface by a set of bi-linear patches corresponding to a mesh with Laplacian smoothing and a mesh with the proposed smoothing method. The experiments show that a mesh smoothed with our method has roughly 20% less approximation error.

Keywords: mesh smoothing, average angle, Laplacian, optimization-based, triangular, quadrilateral

1. INTRODUCTION

This paper presents an angle-based approach to two-dimensional mesh smoothing. For each mesh node our algorithm compares all the pairs of adjacent angles incident to the node and adjusts these angles so that they become equal in the case of a triangular mesh and a quadrilateral mesh, or they form the ideal ratio in the case of a tri-quad mixed mesh. In this paper, we show that this new easy-to-implement smoothing scheme generates a higher quality mesh than the Laplacian smoothing method with less computational cost than the optimization-based smoothing method.

Although automatic mesh generation tools are widely used for finite element analysis, these tools may not guarantee the quality of the meshes. Not only in the process of meshing, but also in mesh refinement and cleanup, it is possible that some severely distorted or ill-shaped elements are created. Even when a uniform mesh is desired, a mesher might generate some elements that are too small or too big compared with the desired element size.

Those mesh quality problems stated above could significantly influence the performance and the accuracy of finite element solutions. There are at least two factors to be considered, the element shape quality and size uniformity. As for the shape quality, for example, a too flat or too thin element in a triangular mesh may increase the approximation error in the finite element solution. For size quality, an excessively coarse mesh can affect the accuracy of the solution, while a too fine mesh is sometimes unnecessary and increases the computational cost.

It is therefore appropriate to perform a post-process to improve the quality of a given mesh in order to improve the performance and the accuracy of the finite element solutions. Such post-processing methods include topological operations and smoothing. Topological operations insert or delete nodes or make edge/face swaps; usually these operations change the topology of the mesh without moving the nodes in the original mesh. Smoothing changes node locations without making any topological changes. We will limit our discussion to the second type of post-process, smoothing, in the rest of this paper.

There are several kinds of mesh smoothing schemes, such as Laplacian smoothing and optimization-based smoothing. Typically each method has a trade-off between quality and computational cost. For example, Laplacian smoothing requires a very low computational cost, but it often results in some low quality mesh elements or even invalid elements. On the other hand, while optimization-based smoothing is more likely to avoid the invalid elements and achieve a higher quality mesh, the computational cost is much higher than Laplacian smoothing.

Our goal is to devise a new mesh smoothing method that strikes a balance between mesh quality and computational cost. The proposed method is applicable to all types of two-dimensional meshes, triangular, quadrilateral and mixed meshes. Our method is stable and effective, guaranteeing a better mesh quality than Laplacian

[*] Correspondence to: Kenji Shimada, Mechanical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890
Phone: (412) 268-3614, Fax: (412) 268-3348

smoothing and avoiding some invalid elements while the computational cost remains much less than optimization-based methods. Furthermore, the algorithm is very easy to implement.

The remainder of this paper is organized as follows: in Section 2, we briefly review the previous work of mesh smoothing, in particular Laplacian smoothing and optimization-based smoothing. Section 3 gives the problem statement along with requirements that we considered in designing our algorithm. The detailed algorithm of our angle-smoothing scheme is presented in Section 4. Some smoothing examples are presented in Section 5.

2. PREVIOUS WORK

There are several methods proposed for mesh smoothing, including Laplacian smoothing [1], optimization-based smoothing [2-7], their combinations [3,5] and some variations of Laplacian smoothing [8-10]. Some general surveys can be found in papers by Owen [11] and Canann *et al* [3]. This section reviews the two most common smoothing schemes, Laplacian smoothing and optimization-based smoothing, used in practical applications.

2.1 Laplacian smoothing

Laplacian smoothing is the most commonly used and straightforward method for mesh smoothing. It simply moves each node to the centroid of the polygon formed by its adjacent nodes. It is a local smoothing algorithm because, in each step, the movement of a node is calculated by using the locations of its adjacent nodes only. We usually run the iteration only a few times because the mesh quality is not improved by further iterations; in fact the quality worsens in many cases.

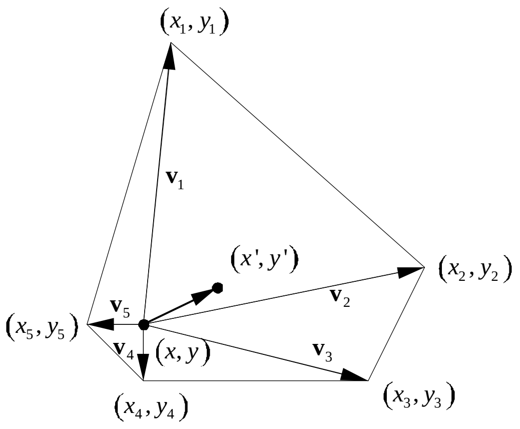


Figure 1. Laplacian smoothing

In Laplacian smoothing we can consider a mesh as a spring system, as shown in Figure 1. Each edge connecting the central node with its neighboring node can be seen as a linear-spring with an initial length of zero. Let $\mathbf{v}_i$ be the vector from the central node to the i th neighboring node**:**

$$\mathbf{v}_i = (x_i - x, y_i - y)$$

The sum of the spring forces acting on the central node is:

$$\mathbf{F} = K \sum_{i=1}^{k} \mathbf{v}_i$$

where K is the spring constant, and k is the number of neighboring nodes. When the central node is located exactly at the geometric center of the polygon, the spring forces are balanced out and the spring system is in equilibrium.

While Laplacian smoothing is an iterative way to find this force-balancing state, we can also solve the problem by minimizing the energy of the spring system. Considering that all the springs have initial lengths of zero, we can compute the potential energy of the system as:

$$E = \sum_{i=1}^{k} \frac{1}{2} K \left(\|\mathbf{v}_i\|_{L_2} \right)^2$$

This is an optimization problem, where the cost function is the above simple quadratic function, as also shown in Figure 2. By minimizing this cost function we can obtain the same results as by Laplacian smoothing.

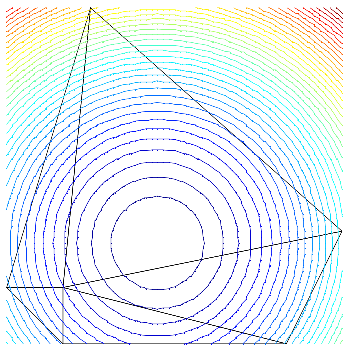


Figure 2. Contour plot of the linear spring energy

Laplacian smoothing can thus be viewed as a kind of optimization of nodal locations. The cost function is the sum of the squared lengths of the edges shared by the same node:

$$f(x, y) = \sum_{i=1}^{k} \left((x - x_i)^2 + (y - y_i)^2 \right)$$

Because the cost function for the Laplacian smoothing method is a simple quadratic function, it becomes very easy to find the node position to minimize this function. We can obtain position (x, y) that minimizes the cost function by simply finding the geometric center of the neighboring nodes:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0$$

$$x = \frac{1}{k}\sum_{i=1}^{k} x_i \,, \; y = \frac{1}{k}\sum_{i=1}^{k} y_i$$

This cost function, however, does not necessarily reflect the mesh quality, and this is the reason why Laplacian smoothing often fails to improve the mesh quality, and sometimes even generates invalid elements.

Below are the advantages and disadvantages of Laplacian smoothing:

Advantages:

- Computationally efficient
- Easy to implement

Disadvantages:

- Does not always move the node to the optimal position to get the best element quality
- May generate inverted elements
- Tends to lose element size uniformity if iterated more than a few times.
- Tends to yield lower quality elements if iterated more than a few times.

2.2 The variations of Laplacian smoothing

In order to overcome some of the disadvantages of Laplacian smoothing, some variations of the algorithm have been proposed. These schemes include:

- Length-weighted Laplacian smoothing [8]

  Although this method uses a set of vectors from an interior node to all its adjacent nodes, the vectors are not simply the vectors from the central node to its adjacent nodes; some weighting factors are applied to these vectors for adjacent nodes on a boundary. This method improves the element quality along the boundary and realizes a rapid convergence speed.

- Surrounding nodes based Laplacian smoothing [9]

  This variation applies to quadrilateral elements only. While the original Laplacian method considers the neighboring nodes connected with the central node, in this variation of Laplacian smoothing, all the surrounding nodes contribute to the displacement of the central node, improving the element shape quality.

- Constrained Laplacian smoothing [3-5]

  In this method, also called “smart” Laplacian smoothing, some mesh quality measures are calculated before and after Laplacian smoothing. If the mesh quality would not be improved by moving a node, the algorithm either does not move the node or moves only a fraction of the calculated amount of displacement. The quality measures used for optimization-based smoothing can also be used here as constraints. This method is effective in avoiding inverted elements. The computational cost, however, is much higher; for example, if the minimum angle is used as the constraint, three angles need to be calculated for each triangular element before and after smoothing.

2.3 Optimization-based smoothing

Optimization-based smoothing methods use some mesh quality measures to define cost functions. Mesh nodes are moved so that the cost function is minimized or maximized.

Some cost functions used in optimization-based smoothing include:

- Minimum/maximum angle [4, 5]

  The minimum/maximum angle is a straightforward index for measuring mesh quality. An element with angles near 0° or 180° will create difficulties in the process of finite element analysis. In optimization-based smoothing, therefore, either the minimum angle is maximized or the maximum angle is minimized in order to eliminate severely distorted elements.

- Aspect ratio [2]

  Aspect ratio is the radius ratio of the circumscribed circle to the inscribed circle of a mesh element. An equilateral triangle has the optimal aspect ratio of 2.0. When the element becomes more distorted, the aspect ratio increases.

- Distortion metrics [3, 12-14]

  The distortion metrics are related to the area and the edge length of elements. The shape quality of an element can be evaluated quantitatively by using this type of metrics. An equilateral triangle has the optimal value, and a severely distorted element has a value near zero. These metrics can also be used for quadrilateral elements.

One advantage of optimization-based smoothing is that it can guarantee the improvement of mesh quality. By optimizing the quality measures, severely distorted elements are effectively eliminated. The computational cost, however, is much higher than Laplacian smoothing. For a two-dimensional triangular mesh, for example, optimization-based smoothing method can be five times more computationally expensive than smart Laplacian smoothing, a variation of Laplacian smoothing [5], and 30 to 40 times more computationally expensive than Laplacian smoothing.

2.4 Hybrid methods

In order to improve the efficiency of smoothing schemes, some approaches are proposed that combine Laplacian smoothing with optimization-based smoothing [3, 5]. The goal of these hybrid methods is to obtain a high quality mesh with relatively less computational cost. One successful approach is to use Laplacian smoothing for most elements for computational efficiency and use optimization-based smoothing only for the poorest quality elements. While the mesh quality of the hybrid methods is not as good as pure optimization-based smoothing, its computational cost is significantly lower.

3. PROBLEM STATEMENT

The smoothing problem we are interested in can be stated as follows:

Input: a two-dimensional triangular, quadrilateral or tri-quad mixed mesh

Output: a smoothed mesh with improved quality

We believe that a good mesh smoothing method should satisfy the following requirements:

- Generality

  A smoothing method should be applicable to triangular, quadrilateral and tri-quad mixed meshes in a consistent manner.

- High quality

  After the smoothing, there should be neither inverted elements nor very poorly shaped elements. It should also work for cases where an input mesh is severely distorted.

  As pointed out by Freitag [4] and Shimada [7], with a severely distorted mesh Laplacian smoothing may result in inverted elements. This flaw can be avoided by adding some constraints, but the mesh quality often cannot be improved further. A good smoothing method should be able to improve the mesh quality while avoiding inverted elements.

- Computational efficiency

  Pure optimization-based smoothing can be too computationally costly to be used in practice. Ideally, a smoothing method has a computational cost comparable to Laplacian or smart Laplacian smoothing.

- Easy implementation

  The Laplacian smoothing method is very straightforward and easy to implement, which is one of the reasons why it has been used widely in practice. Whereas the implementation of an efficient and robust optimization-based smoothing scheme is more complicated.

- Stability and convergence

  We usually run Laplacian smoothing only for a couple of passes and do not use its final convergent result[4]. This is because after the second or the third pass, the improvement becomes very minor, or the mesh quality even degrades. Another reason is that there is a tendency in Laplacian smoothing for a mesh to become less uniform in sizes when we run more passes. This undesirable non-uniformity of element sizes may decrease the accuracy of finite element analysis.

4. ANGLE-BASED SMOOTHING ALGORITHM

This section describes a new smoothing algorithm based on what we call "angle-based smoothing" for triangular, quadrilateral, and tri-quad mixed mesh. The central idea of our method is to make each pair of adjacent angles equal or in a certain ratio, and this is effective in eliminating angles near 0° or 180°. This method is easy to implement, and the quality of a resultant mesh is significantly better than that after Laplacian smoothing, while the computational cost is much lower than optimization-based smoothing.

4.1 Algorithm for triangular mesh

Our angle-based smoothing method can also be considered as a spring system. The difference between this method and the Laplacian method is that the springs used here are torsion springs. The potential energy of such a system of torsion springs is:

$$E = \sum_{i=1}^{2k} \frac{1}{2} K q_i^{\,2}$$

where k is the number of vertices of the polygon, K the spring constant, and q the angle on the boundary of the polygon. Each pair of angles shares one vertex of the polygon; therefore there are $2k$ angles in total to be considered.

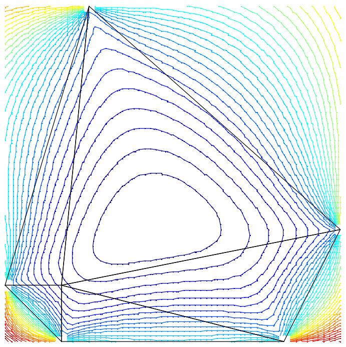


Figure 3. Contour plot of the torsion spring energy

The contour plot of this energy is shown in Figure 3. We use the same polygon as used in Figure 2 for Laplacian smoothing. Comparing these two contour plots we find that the minimum energy positions are different. This position is where the central node will be moved. The angle-based energy optimization gives a better node location than Laplacian's linear spring energy optimization.

We implement an iterative method that minimizes the potential energy of the torsion spring systems. Our angle-based smoothing algorithm is outlined in the following five steps:

1. As shown in Figure 4, for each node N_i, there are k pairs of angles around it, where k is the number of neighboring nodes. The two adjacent angles are calculated as:

$$a_1 = \cos^{-1}\left(\frac{\mathbf{v}_j \bullet \mathbf{v}_{j+1}}{\|\mathbf{v}_j\|\|\mathbf{v}_{j+1}\|}\right)$$

$$a_2 = \cos^{-1}\left(\frac{\mathbf{v}_j \bullet \mathbf{v}_{j-1}}{\|\mathbf{v}_j\|\|\mathbf{v}_{j-1}\|}\right)$$

where $\mathbf{v}_{j-1}$, $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ are the vectors that share vertex N_j; $\|\mathbf{v}\|$ is the L_2 norm of the vector; and a_1, a_2 are the angles determined by the three vectors.

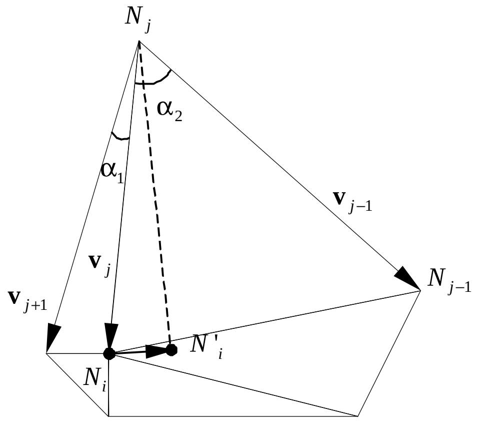


Figure 4. Angle-based method for triangular mesh

2. Calculate the difference between two adjacent angles, and decide the angle for vector $\mathbf{v}_j$ to be rotated:

$$b_j = (a_2 - a_1)/2$$

where b_j is the angle by which vector $\mathbf{v}_j$ will be moved.

3. Rotate vector $\mathbf{v}_j$ by angle b_j about N_j, so the new coordinates of N_i will be:

$$x' = x_0 + (x - x_0)\cos b_j - (y - y_0)\sin b_j$$
$$y' = y_0 + (x - x_0)\sin b_j + (y - y_0)\cos b_j$$

where (x_0, y_0) is the location of node N_j, (x, y) is the old location of node N_i, and (x', y') is the new location of node N_i.

4. By going through all the neighboring nodes, there are k sets of new locations for the same node N_i. We compute the final new location of node N_i by taking average of (x', y') computed from all the neighboring nodes.

$$x_{new} = \frac{1}{k}\sum_{i=1}^{k} x'_i$$

$$y_{new} = \frac{1}{k}\sum_{i=1}^{k} y'_i$$

4.2 Algorithm for quadrilateral mesh

The algorithm for a quadrilateral mesh is similar to that for triangular mesh. The only difference is that we use diagonal nodes in addition to those connected to the central node (see Figure 5). In this case, the ideal angle is 45°, and we can use the same iterative method as for triangular elements to calculate the target angles and then move the node accordingly.

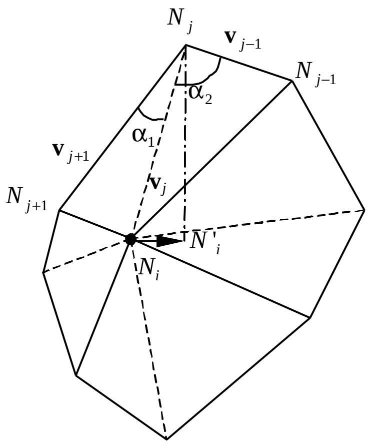


Figure 5. Angle-based method for quadrilateral mesh

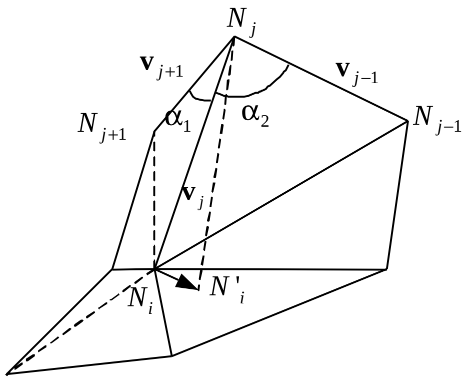


Figure 6. Angle-based method for tri-quad mixed mesh

4.3 Algorithm for tri-quad mixed mesh

In case of smoothing a tri-quad mixed mesh, because the ideal angles for triangular elements and quadrilateral elements are different we need to divide the two adjacent angles into a particular ratio instead of finding the bisection of the angles. We use the ratio of $90° : 60° = 3 : 2$; as shown in Figure 6; suppose that a_1 is an angle from a quadrilateral element, and a_2 from a triangular element, the target ratio should be $a_1 : a_2 = 3 : 2$.

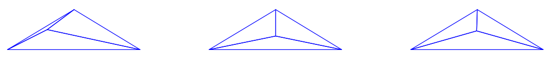
5. RESULTS AND DISCUSSION

5.1 Quality improvement

Both the shape quality and the size quality are important for finite element analysis, for they can affect the accuracy and efficiency of the analysis. The following results and discussion highlight the advantage of our angle-based method by showing how our method can improve the quality of different types of meshes: topologically irregular meshes, severely distorted meshes, uniform meshes and graded meshes.

5.1.1 Special cases handling

The first special case can be often found on the boundary of a mesh. Such an example is shown in Figure 7. If Laplacian smoothing is used, the element at the bottom becomes flat. Whereas, the angle-based method improves the shape of this element significantly.

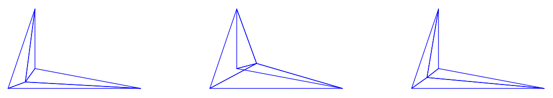


(a) Original mesh	(b) Laplacian smoothing	(c) Angle-based smoothing
Min angle=4.40°	Min angle=11.31°	Min angle=15.48°
Max angle=169.70°	Max angle=157.38°	Max angle=149.04°

Figure 7. The original topologically irregular mesh, the same mesh after Laplacian smoothing and average angle smoothing

One disadvantage of Laplacian smoothing is that it occasionally generates invalid mesh elements. The example shown in Figure 8 illustrates how Laplacian smoothing can lead to such elements by moving the center vertex out of the boundary, yielding an invalid mesh for finite element analysis. We could choose not to move this node to avoid the inverted elements, as done in a smart Laplacian smoothing, but it does not improve the quality of the original mesh at all.

By using the angle-based smoothing, we can effectively avoid the inverted element while improving the quality of the mesh as can be seen from Figure 8(c).



(a) Original mesh (b) Laplacian smoothing (c) Angle-based smoothing

Figure 8. A comparison between Laplacian smoothing and angle smoothing for severely distorted mesh

5.1.2 Shape improvement

In this sub-section, we present two examples of mesh smoothing, one with a triangular mesh and the other with a tri-quad mixed mesh. For the triangular mesh example, we placed 500 nodes in a square domain, and 20 equally spaced nodes on each edge of the boundary. We used randomly placed nodes and then triangulated them using the Delaunay method, yielding 580 nodes and 1078 triangular elements in the domain. Also for the mixed mesh example, we used a similar method to generate the mesh, and there are 580 nodes, 262 triangular elements and 408 quadrilateral elements in total.

We implemented Laplacian smoothing, smart Laplacian smoothing and angle-based smoothing to compare the mesh quality improvement. In smart Laplacian smoothing, we used the minimum angle as the constraint function. Tables 1 and 2 list the numerical results of quality improvement. The meshes are shown in Figures 9 and 10.

For the triangular mesh, smart Laplacian method can produce a higher quality mesh than Laplacian smoothing. For the quad mesh, although smart Laplacian improves the minimum angles better, it could not improve the maximum angle and even generated concave elements. Perhaps we could avoid the concave elements by using other constraints, but it would be more computationally expensive.

In our angle-based method, we use the same scheme for triangular and quad meshes, described in Section 4, and the results are better than Laplacian and smart Laplacian smoothing.

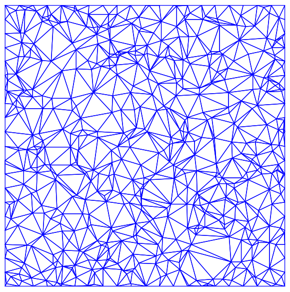
Table 1. Mesh quality improvement for triangular mesh

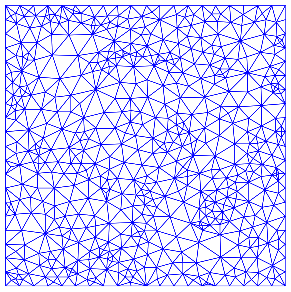
Case	Min angle	Max angle	(a)	(b)	(c)
Original	0.061°	179.76°	281	48	6.05E10^3
Laplacian	9.99°	156.20°	13	5	86
Smart Laplacian	11.85°	154.06°	11	6	75
Angle-based	20.01°	136.32°	0	1	17

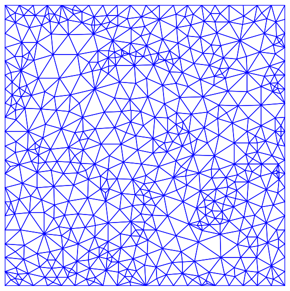
(a): Number of angle < 20°

(b): Number of angle >130°

(c): The ratio of minimum area and maximum area of the element


(a) Original mesh


(b) Laplacian smoothing


(c) Smart Laplacian smoothing

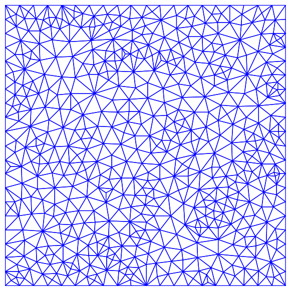
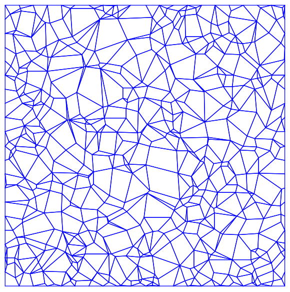
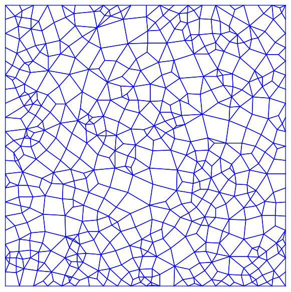
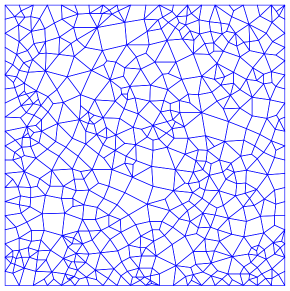

(d) Angle-based smoothing

Figure 9. Example for triangular mesh smoothing


(a) Original mesh


(b) Laplacian smoothing


(c) Smart Laplacian smoothing

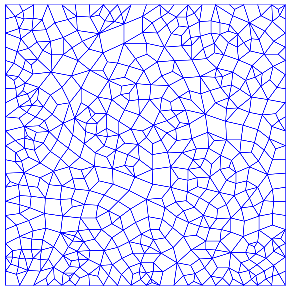
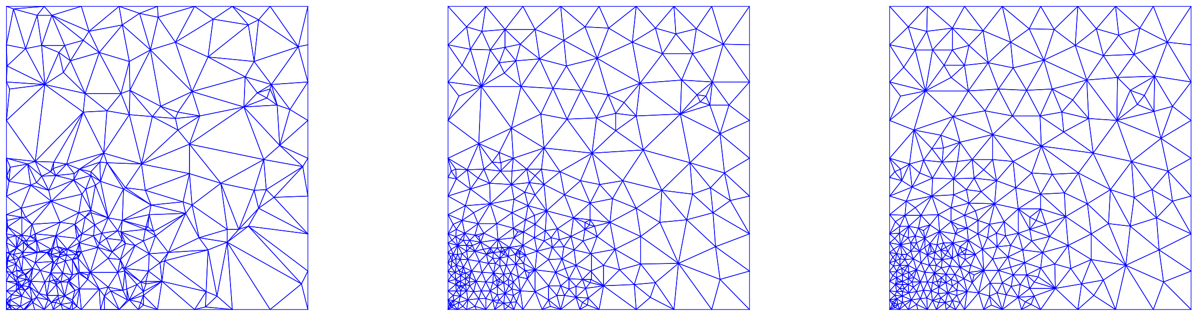

(d) Angle-based smoothing

Figure 10. Example for tri-quad mixed mesh smoothing



(a) Original mesh **(b) Laplacian smoothing** **(c) Angle-based smoothing**

Figure 11. Example of smoothing graded mesh

Table 2. Mesh quality improvement for tri-quad mixed mesh

	Min angle	Max angle	(a)	(b)
Original	1.37°	176.56°	116	37
Laplacian	8.82°	174.02°	13	13
Smart Laplacian	11.44°	226.28°	4	32
Angle-based	19.17°	155.57°	4	5

(a): Number of angle < 20°

(b): Number of angle >150°

5.1.3 Size improvement

Controlling element sizes is also important for finite element analysis. There are two cases that we should consider: (1) an original mesh is intended to be uniform, and (2) an original mesh is intended to be graded. For a uniform mesh, a too coarse element may decrease the accuracy of the result, while a too fine element is often unnecessary and increase the computation cost. Therefore, the size uniformity is needed for this kind of mesh. For a graded mesh, we prefer uniformly sized elements in the same layer, and the size change at the transitional zone should be smooth.

The uniform mesh examples can be found in Figures 9, 10 and 15. The minimum and maximum element areas were calculated and shown in Table 3 for triangular and quadrilateral mesh examples. The angle-based smoothing can reduce the undesirable difference between the minimum and maximum areas more than Laplacian smoothing.

Table 3. Mesh size improvements for 2D uniform meshes

	Original		Laplacian		Angle-based	
	Min	Max	Min	Max	Min	Max
Triangular	0.01	60.36	0.35	35.56	1.42	24.17
Quadrilateral	1.55	197.95	2.95	126.35	8.54	109.02

Another example is a graded mesh shown in Figure 11. The mesh was generated by placing 300 random nodes in a square domain. In order to make the gradation, the first 100 nodes were distributed in the whole domain, the second 100 nodes were distributed only in the lower-left quarter of the domain, and the last 100 nodes were distributed only in the lower-left 1/16 of the domain. So there are three layers in this graded mesh. We also placed 48 fixed nodes on the four edges of the domain. All the nodes were connected by Delaunay triangulation.

From the result shown in Figure 11, we can see that the angle-based method yields a better result than Laplacian smoothing in terms of element size control. In a same layer, the element sizes tend to be more uniform; and in a transitional zone, the size change is smoother with angle-based smoothing than with Laplacian smoothing.

5.2 Computational cost

The most notable advantage of Laplacian smoothing is the computational efficiency. For a same mesh, optimization-based smoothing might take 30 or 40 times more computational time than Laplacian smoothing [4]. Some researchers [3, 5] use smart Laplacian to improve the performance of Laplacian smoothing. The computational cost of those variants of Laplacian smoothing usually falls between the computational costs of Laplacian and optimization-based smoothing [4].

To compare the computational cost between our angle smoothing and other smoothing methods, we estimated the computational cost of a kind of smart Laplacian smoothing method and optimization-based smoothing method, and compared them with our angle smoothing method. The algorithm of the smart Laplacian method is according to Freitag [5], and we use the minimum angle as the mesh quality measure:

1. For each node, calculate all the angles in its adjacent elements. In a topologically regular triangular mesh, for example, there are 6 adjacent elements, and therefore there exist 18 angles to be computed.
2. From the set of angles, find the minimum angle.
3. Compute the new location of the central node using Laplacian smoothing.
4. Calculate the 18 angles again.
5. Find the new minimum angle.
6. If the minimum angle is improved, move the central node to the new location. Otherwise keep the node at its original location.

In this algorithm, there are totally 36 angles to be calculated before and after one node is moved. The angle calculation can be simplified to getting the cosine value of each angle, which can save some computational time, but this angle calculation is still the most time-consuming part in the overall smoothing algorithm.

In our angle-based smoothing algorithm, if the number of adjacent elements is 6, there are 12 angles to be calculated, and 6 vector rotations. Another more efficient way is to find the bisection of the angle by normalizing the two vectors.

The computation cost of optimization-based smoothing is usually higher than smart Laplacian. For example, some researchers [3, 5] used steepest descent method and move nodes in gradient directions so as to increase mesh quality. If we use the angle to measure the mesh quality, there are still 36 angles to be calculated before and after the movement. In addition, the gradient directions need to be computed.

From the comparison above, we can see that, in every pass, angle-based smoothing method is faster than smart Laplacian and optimization-based smoothing method. If we run the angle-based smoothing only 2 or 3 passes, the mesh quality is already better than Laplacian smoothing; and if we run more iterations, the quality will improve further. In the next sub-section, the convergence speed of angle-based smoothing is discussed and compared with that of Laplacian smoothing.

5.3 Convergence

In Section 4 we stated that our angle-based smoothing is an iterative method of finding the optimal node positions that minimizes the potential energy in the torsion spring system. In order to investigate the convergence of our algorithm, we use the same 5-element shown in Figures 2 and 3. Three different initial positions are given for the central node, and our angle-based smoothing is applied. We iterated the smoothing process 10 times and the converging paths are shown in Figure 12.

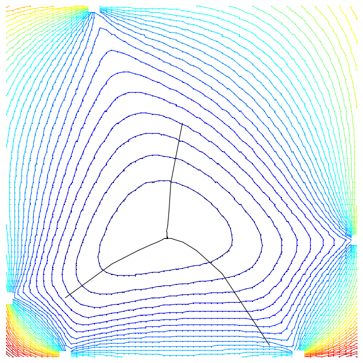


Figure 12. Converging path of the central node from 3 different initial conditions

During the iteration, the displacement of the central node decreases exponentially; the amount of the displacement of the 10th pass is 0.1% of that of the first pass. The angle-based smoothing converges fast and the final solution is independent of the initial conditions.

As discussed in Section 3, it is important to decide how many passes, or iterations, we need to improve a mesh. Table 4 shows the effect of the number of passes on mesh quality improvement in Laplacian smoothing and our angle smoothing.

Table 4. Effect of the number of passes for Laplacian and angle-based smoothing

Passes	Laplacian			Angle-based		
	Min angle	Min area	Max area	Min angle	Min area	Max area
1	8.84°	1.69	56.84	8.70°	1.40	58.75
2	11.45°	1.71	59.82	12.25°	2.85	56.52
3	10.88°	1.64	61.04	13.56°	3.45	55.34
4	10.93°	1.60	61.61	14.20°	3.69	54.69
5	11.01°	1.59	61.89	14.91°	3.90	54.32
6	11.07°	1.59	61.96	15.63°	4.09	54.12
7	11.12°	1.61	61.94	16.33°	4.22	54.01
8	11.17°	1.62	61.85	17.01°	4.31	53.96
9	11.20°	1.64	61.74	17.65°	4.37	53.94
10	11.23°	1.65	61.62	18.25°	4.42	53.95

Figure 13 illustrates how the maximum nodal displacement, minimum angle, and the ratio of minimum area and maximum area of elements change as we perform more passes. From the displacement curves shown in the top of Figure 13, we can see that both methods become stable after some passes. It should be noted in the middle of Figure 13, however, that Laplacian smoothing maximizes the minimum angle after the second pass, and the angle becomes slightly worse after that. On the other hand our angle-based smoothing keeps improving the minimum angle as more passes are performed. Also note that the curves shown in the bottom of Figure 13 indicate that our method generates more uniformly sized mesh elements than Laplacian smoothing.

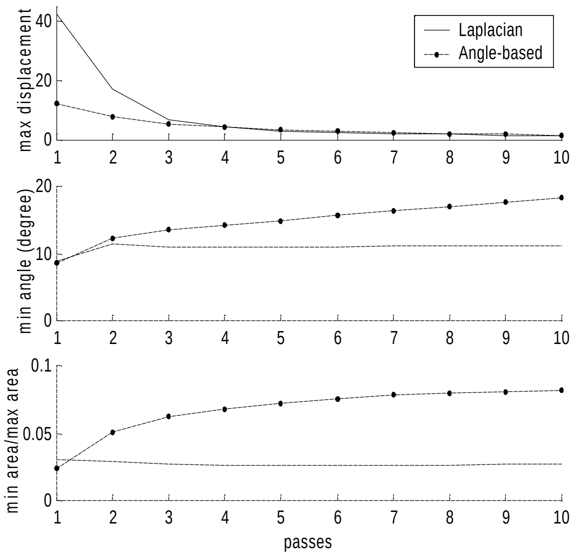


Figure 13. Comparison of Laplacian and angle-based smoothing by number of passes

5.4 Application to surface approximation

In order to prove the effectiveness of the proposed angle smoothing method, we compare errors in approximating a give analytical surface by a set of bi-linear patches corresponding to a mesh with Laplacian smoothing and a mesh with our method[15].

Assuming that a surface is defined over a square domain meshed into a set of quadrilateral elements the surface is approximated by a set of bi-linear patches. Each quadrilateral element has a corresponding bi-linear patch that approximates the original surface geometry.

Suppose $f(x,y)$ is the actual value of the surface at point(x,y), $f'(x,y)$ is the interpolated value obtained by bi-linear interpolation. Thus the error at the point (x,y) is:

$$E(x,y) = f(x,y) - f'(x,y)$$

For each element, the square of the errors are summed and weighted by the element area to quantify the amount of surface approximation errors:

$$\|E\|_{L_2} = \left[\int_{\Omega} E^2 dA\right]^{\frac{1}{2}}$$

The total approximation error, which we will use to compare the performance of our method with the Laplacian method, is the summation of the errors computed for each quadrilateral element.

We use the following three functions as test cases.

$$f_1(x,y) = \sin\left(\frac{2p\,x}{l}\right)\sin\left(\frac{2p\,y}{l}\right)$$

$$f_2(x,y) = \sqrt{62500 - x^2 - y^2}$$

$$f_3(x,y) = x^3$$

The surface plots are shown in Figure 14.

The original mesh is a quadrilateral mesh with 571 nodes and 530 quadrilateral elements. We made a comparison between the approximation errors using a Laplacian smoothed mesh and an angle-smoothed mesh.

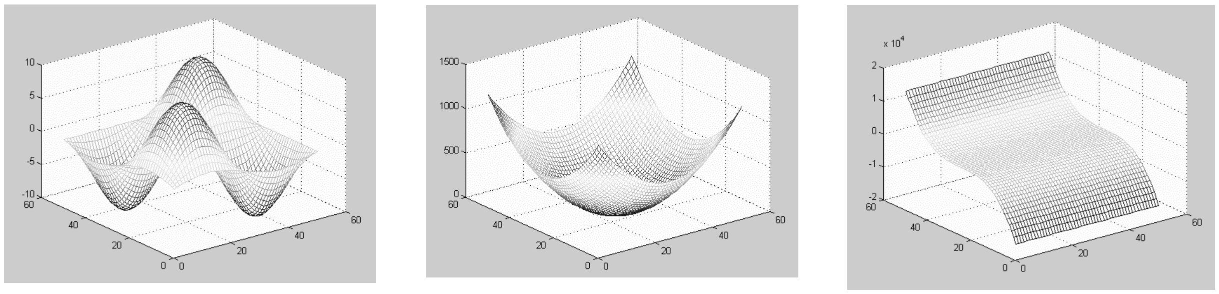
Table 5 summarizes the numerical results of the approximation errors. As can been seen from these results a mesh smoothed with our angle-based method typically has 20% less approximation error than a Laplacian mesh. Note that this improvement is achieved by moving existing nodes to better locations without changing the topology or increasing the number of elements.

This result indicates that our method generates more uniform mesh elements over a domain, and it helps improve the surface approximation errors, which is analogous to the errors in finite element solutions incurred by polynomial interpolation of a true solution.

It should be noted that we are not adapting a mesh based on an error estimator or an estimation of the curvature of a surface. Such an adaptive remeshing scheme would improve the approximation error more dramatically, but we would need some knowledge about the surface geometry. Our interest here is to see how smoothing algorithms alone can improve the approximation error without adapting a mesh based on knowledge of the surface geometry.

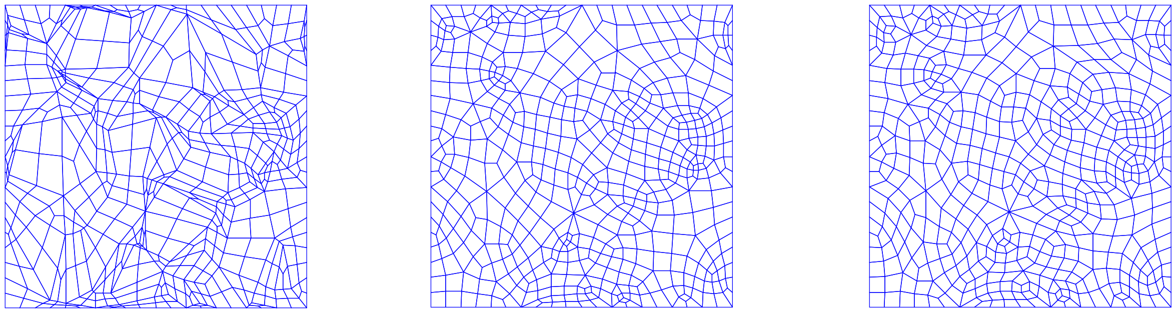
Table 5. Comparison of the surface approximation errors with Laplacian smoothing and angle smoothing

Function	Laplacian	Angle-based	Error Improvement
f_1	95.38	77.08	23.74%
f_2	0.23	0.19	21.05%
f_3	1.16E8	1.00E8	16.00%



(a) The surface plot of function 1 (b) The surface plot of function 2 (c) The surface plot of function 3

Figure 14. The surface plots of the application functions



(a) Original mesh **(b) Laplacian smoothing** **(c) Angle-based smoothing**

Figure 15. Application example for quadrilateral mesh

5 CONCLUSION

In this paper we proposed a new angle-based mesh smoothing algorithm that works for all types of two-dimensional meshes: triangular, quadrilateral and tri-quad mixed meshes.

By adjusting the angles of adjacent elements and imposing a proper angle ratio between adjacent angles, our method generates a higher quality mesh than Laplacian smoothing while keeping the computational cost much lower than optimization-based methods.

We also presented the physically-based interpretation of our angle-based smoothing as well as that of Laplacian smoothing. This give us insights explaining why Laplacian-based smoothing methods do not always improve the mesh quality, and why our angle-based approach works better.

Although we discussed only two-dimensional mesh smoothing in this paper, the same angle-based idea could be applied for three-dimensional mesh smoothing.

ACKNOWLEDGEMENTS

The material is based in part on work supported under a NSF CAREER Award (No. 9985288) and a grant from IBM Research.

We are grateful for the insightful technical discussions with A. Yamada, T. Itoh and K. Inoue of IBM Research in the development of this work.

REFERENCES

[1] D. A. Field, “Laplacian Smoothing and Delaunay Triangulation,” *Communications in Applied Numerical Methods*, vol. 4, pp. 709-712, 1988.

[2] V. Parthasarathy and S. Kodiyalam, “A constrained optimization approach to finite element mesh smoothing,” *Finite Elements in Analysis and Design*, vol. 9, pp. 309-320, 1991.

[3] S. A. Canann, J. R. Tristano, and M. L. Staten, “An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes,” presented at *7th International Meshing Rountable*, 1998.

[4] L. Freitag and C. Ollivier-Gooch, “a Comparison of Tetrahedral Mesh Improvement Techniques,” presented at the 5th International Meshing Roundtable, Albuquerque NM, 1995.

[5] L. A. Freitag, “On Combining Laplacian and Optimization-based Mesh Smoothing Techniques,” *AMD Trends in Unstructured Mesh Generation*, vol. 220, pp. 37-43, 1997.

[6] L. Freitag, M. Jones, and P. Plassmann, “a Parallel Algorithm for Mesh Smoothing,” *Society for Industrial and Applied Mathematics*, vol. 20, pp. 2023-2040, 1999.

[7] K. Shimada, “Physically-based mesh generation: Automated triangulation of surfaces and volumes via bubble packing,” . Cambridge, MA: Massachusetts Institute of Technology, 1993.

[8] T. D. Blacker, “Paving: a New Approach to Automated Quadrilateral Mesh Generation,” *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 811-847, 1991.

[9] J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, and J. Wu, “a New Approach to the Development of Automatic Quadrilateral Mesh Generation,” *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 849-866, 1991.

[10] P. Hansbo, “Generalized Laplacian Smoothing of Unstructured Grids,” *Communications in Numerical Methods in Engineering*, vol. 11, pp. 455-464, 1995.

[11] S. Owen, “a Survey of Unstructured Mesh Generation Technology,” *www.andrew.cmu.edu/user/sowen/mesh.html*

[12] S. H. Lo, “a New Mesh Generation Scheme for Arbitrary Planar Domains,” *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1403-1426, 1985.

[13] S. H. Lo, “Generating Quadrilateral Elements On Plane and Over Curved Surfaces,” *Computers & Structures*, vol. 31, pp. 421-426, 1989.

[14] M. Berzins, “Mesh Quality: A Function of Geometry, Error Estimates or Both?,” presented at 7th International Meshing Roundtable, Dearborn, Michigan, 1998.

[15] N. Viswanath and K. Shimada, “Adaptive Anisotropic Quadrilateral Mesh Generation Applied to Surface approximation,” . Pittsburgh, PA: Carnegie Mellon University, 2000.