2D FE Quad Mesh Smoothing via Angle-Based Optimization

Hongtao Xu and Timothy S. Newman

Department of Computer Science, University of Alabama in Huntsville,
Huntsville, Alabama 35899
{hxu, tnewman}@cs.uah.edu

Abstract. A new mesh smoothing algorithm that can improve quadrilateral mesh quality is presented. Poor quality meshes can produce inaccurate finite element analysis; their improvement is important. The algorithm improves mesh quality by adjusting the position of the mesh's internal nodes based on optimization of a torsion spring system using a Gauss-Newton-based approach. The approach obtains a reasonably optimal location of each internal node by optimizing the spring system's objective function. The improvement offered by applying the algorithm to real meshes is also exhibited and objectively evaluated using suitable metrics.

1 Introduction

Finite element (FE) analysis acts on mesh elements that are usually generated by applying mesh generation algorithms. Accurate FE analysis results depend on the mesh being valid (i.e., having valid elements), having no slivers, and conforming to the given domain's boundary. It is also best if the mesh's density varies smoothly.

Meshes generated by mesh generation algorithms can often be optimized with a mesh smoothing algorithm. A smoothing algorithm relocates nodes so that the mesh will be of a higher quality. Mesh smoothing is usually done in an iterative process that does not change element connectivity. One popular 2D mesh smoothing algorithm is the Laplacian Smoothing Algorithm [1]. Laplacian smoothing often produces satisfactory smoothing results. However, it can sometimes generate meshes with sliver-like elements or with invalid elements.

In this paper, we consider an alternate smoothing algorithm, Zhou and Shimada's [2] torsion spring-based algorithm, that is better at avoiding generation of slivers and invalid elements. The Zhou and Shimada algorithm can be implemented easily, but it does not optimally utilize the torsion spring system it is based on. The new smoothing approach presented here smooths a mesh of quadrilateral elements in a manner that is provably optimal for the torsion spring formulation proposed by Zhou and Shimada.

This paper is organized as follows. Section 2 discusses related work. Section 3 introduces the new algorithm. Section 4 presents results and an evaluation of the algorithm. Section 5 presents the conclusion.

2 Previous Work

A number of mesh smoothing methods for producing acceptable quality meshes have been presented previously (e.g., [1,3,4,5,6,7,8]), including approaches that minimize a distortion metric (e.g., as in [3]), that disconnect invalid elements from the remaining mesh (e.g., as in [4]), that solve a generalized linear programming problem (e.g., as in [5]), that combine constrained Laplacian smoothing together with an optimization-based smoothing algorithm (e.g., as in [1,6]), that use parallel algorithms (e.g., as in [7]), and that generalize Laplacian smoothing (e.g., as in [8]). In this section, we describe the Laplacian smoothing and Zhou and Shimada's smoothing, which are relevant to our work.

The Laplacian Smoothing Algorithm [9] is an iterative method. It is widely used due to its relative efficiency and simplicity in application, although it may generate slivers or invalid elements in some cases. In 2D meshes, the Laplacian Smoothing Algorithm attempts to improve mesh quality by moving internal mesh nodes in one or more smoothing passes. In each pass, it moves each internal node to the centroid $(\bar{x}, \bar{y})$ of the polygon about the internal node. By polygon about the internal node, we mean the polygon whose vertices are the nodes connected to the internal node.

Zhou and Shimada [2] have presented a physically-based mesh smoothing algorithm. It accomplishes smoothing by moving each internal node to a better location based on modeling the edges that connect nodes as a torsion spring system. For a set of nodes connected to an internal node, if the edges that connect these nodes to the internal node are viewed as a torsion spring system, then this torsion spring system's energy can be expressed as:

$$E = \sum_{i=0}^{2(n-1)} \frac{1}{2} k {\theta_i}^2, \tag{1}$$

where n is the number of nodes that are connected to the internal node by an edge, k is a constant, and θ_i is the angle between a polygon edge and the line from the internal node to the *i-th* connected node.

Zhou and Shimada's algorithm minimizes the energy of the torsion spring system (i.e., that was shown in Eqn. 1) and then uses a heuristic to relocate the interior nodes of the mesh. More detail can be found in [2].

3 Angle-Based Optimization Smoothing

As described earlier, the Laplacian smoothing can generate slivers and invalid elements. Other mesh smoothing methods, such as optimization-based methods including the Zhou and Shimada algorithm, can often give better quality meshes than Laplacian smoothing, especially for meshes that contain badly shaped finite elements. However, the Zhou and Shimada algorithm uses a non-optimal heuristic. It is possible to instead use an optimization approach to optimally relocate each internal node. In this section, our optimization approach that accurately minimizes the energy of the torsion spring system to produce a well-smoothed quadrilateral mesh is presented. The approach better-optimizes the new locations of all internal nodes. The approach is an extension of our earlier work on triangular meshes [10].

3.1 Objective Function for Optimal Angle

The energy of a torsion spring system model on the edges that connect nodes is provably minimal when the angle at each vertex is divided by a bisecting line. However, the bisectional lines for a polygon about an internal node seldom intersect at the same point. Zhou and Shimada's heuristic estimates an internal node's new location using averaging, which is not optimal. To find the optimal new location of nodes, our approach uses the following objective function s:

$$s = \sum_{i=0}^{n-1} [distance(\hat{D}', L_i)]^2, \tag{2}$$

where L_i is the bisectional line of the internal angle at node D_i, D_i is one node of the polygon about an internal node $\hat{D}$, and where $\hat{D}'$ is the new (i.e., optimized) position of the internal node.

We can define $t = (p, q)$ as the coordinate of an internal node. Since the function $distance(\hat{D}', L_i)$ is a function of the coordinates of an internal node, the distance can be expressed as a function f_i of the coordinate t, allowing the objective function s in Eqn. 2 to be re-written as:

$$s = \sum_{i=0}^{n-1} f_i(t)^2. \tag{3}$$

The least squares formulation for an objective function s is:

$$min \quad s(t) = \frac{1}{2} f(t)^T f(t), \tag{4}$$

where $t = (t_{1,} t_{2,} ..., t_n)$ is a vector with n components, $f(t)$ is a column vector of m functions $f_i(t)$ and $f(t)^T$ is the row vector $f(t)^T = (f_1(t), ..., f_m(t))$.

The formulation of Eqn. 4 can be solved by any optimization that minimizes the objective function $s(t)$. When $m = n$, this problem becomes a set of non-linear equations in the form $f(t) = 0$.

3.2 Derivation of Linear Equation System

By taking the derivative of s in Eqn. 4, the following equation can be obtained:

$$P = \frac{\partial s}{\partial t} = \sum_{i=1}^{n} f_i \frac{\partial f_i}{\partial t} = \frac{\partial f^T}{\partial t} f = J^T f, \tag{5}$$

using $J^T = \frac{\partial f^T}{\partial t}$,where J^T is the transpose of the Jacobi matrix, and using $P = \left(\frac{\partial s}{\partial t_1} \; \frac{\partial s}{\partial t_2} \; ... \; \frac{\partial s}{\partial t_n} \right)^T$. Thus, the Jacobi matrix can be expanded as:

$$J = \begin{pmatrix} J_{11} & \dots & J_{1n} \\ \vdots & \dots & \vdots \\ J_{m1} & \dots & J_{mn} \end{pmatrix}, \tag{6}$$

where $J_{ij} = \frac{\partial f_i}{\partial t_j}$.
To minimize the objective function in Eqn. 4, we should have $P = 0$. By expanding P at point $t = t^n$ with a Taylor series that deletes derivatives of second and above order, we obtain

$$P(t) = P(t^n) + \left.\frac{\partial P}{\partial t}\right|_{t^n} (t - t^n). \tag{7}$$

Then, substituting $P = 0$ into Eqn. 7, we can obtain

$$t = t^n - (\left.\frac{\partial P}{\partial t}\right|_{t^n})^{-1} P(t^n). \tag{8}$$

Clearly,

$$\left.\frac{\partial P}{\partial t}\right|_{t^n} = \left.\frac{\partial (J^T f)}{\partial t}\right|_{t^n}. \tag{9}$$

Deleting the derivatives of second and above order, as is done in Eqn. 7, we obtain

$$\left.\frac{\partial P}{\partial t}\right|_{t^n} = {J_n}^T J_n, \tag{10}$$

where J_n is the value of J at $t = t^n$.
By substituting Eqns. 10 and 5 into Eqn. 8, and using t^{n+1} as the next step value, the following equation can be obtained:

$$t^{n+1} = t^n - ({J_n}^T J_n)^{-1} {J_n}^T f_n. \tag{11}$$

Next, we define

$$d^n = -({J_n}^T J_n)^{-1} {J_n}^T f_n, \tag{12}$$

which allows Eqn. 11 to be simplified to

$$t^{n+1} = t^n + d^n. \tag{13}$$

In Eqn. 13, it is possible to prove that d^n is always negative. Thus, t^n decreases with increasing n [11]. Although d^n can be very small, there are still chances that a better point exists in the interval $[0, d^n]$.To further optimize Eqn. 13 in the interval, a scalar λ^n with domain $[0, 1]$ can be introduced into Eqn. 13, leading to the following equation:

$$t^{n+1} = t^n + \lambda^n d^n. \tag{14}$$

Iterative search for a solution t is then used until a specified precision is reached. In each iteration, a one dimensional search of λ is made. The solution t can be substituted into Eqn. 4 to find an optimal s, solving the problem.

3.3 Optimization Algorithm

Next, we describe our approach's use of Gauss-Newton optimization to optimize the objective function s. Optimization of s requires finding vector t. To solve Eqn. 3, it is necessary to find the optimal λ^n in Eqn. 14. Therefore, the following problem, where only λ^n is unknown, needs to be solved:

$$min\ s(t^n + \lambda^n d^n). \tag{15}$$

To solve this problem, the following steps are used. These steps require the solution's precision ϵ to be pre-specified. That is $\left|\frac{s(t^n+d^n)-s(t^n)}{s(t^n)}\right| < \epsilon$.

```
Step 1. Calculate $s(t^n + d^n)$ and $s(t^n)$.
Step 2. If solution's $\epsilon$ has been reached or it is iteration $j_{\max}$, go to Step 8.
Step 3. Set $\lambda^n = 1$.
Step 4. If $s(t^n + d^n) < s(t^n)$, set $t^n = t^n + d^n$ and go to Step 1.
Step 5. Assume $s(t^n + \lambda^n d^n)$ with respect to $\lambda^n$ is quadratic and find the co-
        efficients of the quadratic polynomial using the values of $s$ at $\lambda^n = 0$,
        $\lambda^n = 1$, and the derivative of $s$ at $\lambda^n = 0$.
Step 6. Find minimum value of $s(t^n + \lambda^n d^n)$ for $0 \leq \lambda^n \leq 1$.
Step 7. a. If $s(t^n + d^n) < s(t^n)$, go to Step 1.
        b. Set $\lambda^n = \lambda^n/2$.
        c. Set $t^n = t^n + \lambda^n d^n$.
        d. Go to Step 7a.
Step 8. Stop.
```

In practice, we have used 10 iterations ($j_{max} = 10$) which has led to reasonable solutions.

4 Results and Discussion

In this section, the qualitative characteristics and computational results of the new smoothing algorithm are presented. The algorithm's characteristics and performance are also compared with the Laplacian smoothing algorithm.

The comparison uses mesh angle and length ratios. An element's angle ratio is the ratio of the minimum angle (of the four angles of an element) to the maximum angle (of the four angles of the same element). The length ratio is the ratio of an element's minimum side length to its maximum side length.

Metrics that are derived from the angle and length ratios are used to evaluate mesh quality. The basic derived metric is the *metric ratio*. The metric ratio is the product of the angle ratio and the length ratio.

The *element area* is also used to determine quality. The metrics derived from this measure and used in evaluation are the maximum and minimum element areas for the whole mesh.

Next, we report experiments that test mesh quality for three scenarios, which we call Cases 1, 2, and 3.

Shape Improvement. The meshes for the Case 1, 2, and 3 scenarios are shown in Figure 1 (a), (b), and (c), respectively. Figure 1(d), (e), and (f) show the meshes created by applying Laplacian smoothing to the original mesh. Figure 1(g), (h), and (i) show the meshes created by applying the new smoothing algorithm to the original mesh. In all three cases, the meshes produced by both Laplacian smoothing and the new smoothing appear to be more uniform in shape than the original mesh. The mesh quality metrics for the scenarios are shown in Table 1. In this table, the worst metric ratios of the original mesh, the mesh generated by Laplacian smoothing, and the mesh generated by the new smoothing algorithm are shown. The metric values for Laplacian smoothing are all greater than the metric values for the original mesh, which means that the overall mesh uniformity in shape is improved by Laplacian smoothing. In particular, the larger worst metric value means that the worst elements are better in the mesh smoothed by the Laplacian algorithm than they are in the original mesh. The new smoothing algorithm's worst metric values are greater than the metric values for Laplacian smoothing in all three cases, which means that the new algorithm produced a mesh with a less extreme worst element.

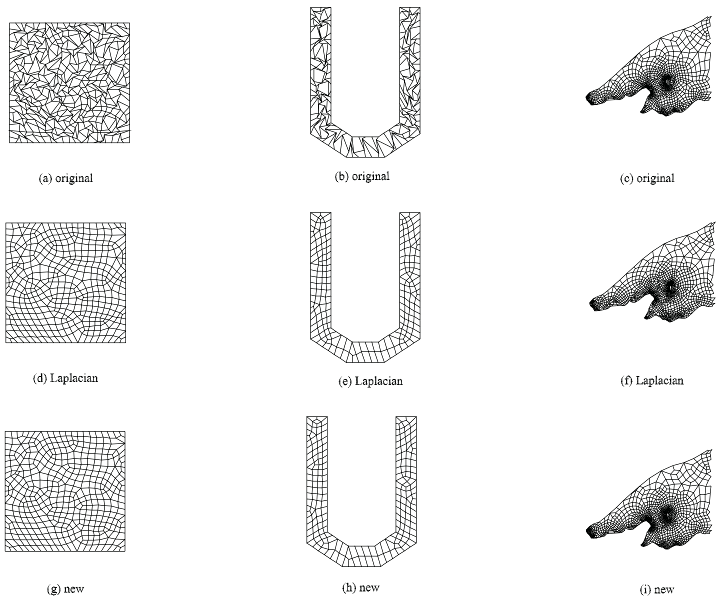

Fig. 1. Case 1, 2, 3 meshes and results of smoothings on them

Table 1. Mesh Quality Metrics for Case 1, 2 and 3 Scenario

	Metric Name	Original Mesh	Laplacian	New Smoothing
Case 1	Worst metric ratio	0.0017	0.0586	0.0591
Case 2	Worst metric ratio	0.0011	0.0793	0.0912
Case 3	Worst metric ratio	0.0019	0.0111	0.0127

In summary, the new algorithm appears to produce reasonable meshes whose worst elements are of a much higher quality than in the Laplacian smoothing; the new algorithm's worst elements are less sliver-like.

Size Improvement. Minimum and maximum element areas are used to measure the size uniformity for the quadrilateral meshes produced by Laplacian smoothing and the new smoothing. Table 2 reports these metrics for the original mesh, the mesh generated with Laplacian smoothing and the mesh generated with the new smoothing for the three cases discussed here. With the exception of the Case 2 scenario minimum area, the new smoothing's meshs' minimum areas are all greater than the minimum areas from Laplacian smoothing. The maximum areas for the new smoothing are also all less than the maximum areas from Laplacian smoothing. More importantly, the variation in extreme size is lowest for the new algorithm. Thus, the meshes generated by the new smoothing algorithm tend to have less extreme element sizes. While our examples demonstrate that Laplacian smoothing can improve mesh quality, it can be further seen that the new smoothing algorithm can generate a mesh with less extreme variation in element size than does Laplacian smoothing.

Table 2. Mesh Quality Metrics for 3 Scenarios

		Original Mesh	Laplacian	New Smoothing
Case 1	Min. area	0.0251	0.0516	0.0676
	Max. area	0.6954	0.4405	0.4125
Case 2	Min. area	2.8810	6.9078	6.8607
	Max. area	94.979	50.102	44.036
Case 3	Min. area	0.0141	0.0285	0.0326
	Max. area	14.960	14.199	12.967

5 Conclusion

In this paper, a new smoothing algorithm for quadrilateral mesh smoothing has been presented. The new smoothing algorithm uses an angle-based optimization method to optimize a torsion spring system. The formulation is set up to optimize the locations of all internal nodes. The solution is found based on the Gauss-Newton optimization.

Here, the new mesh smoothing algorithm's performance in reducing sliver elements was also reported. The testing results lead to the following conclusions. First, the new smoothing algorithm produces better quadrilateral element shapes than does the Laplacian Smoothing Algorithm. Furthermore, the new algorithm gives better mesh uniformity than that generated with Laplacian smoothing.

References

1. L. Freitag: On Combining Laplacian and Optimization-Based Mesh Smoothing Techniques. In: Proc., 6th Int'l Mesh. Roundtable, AMD-Vol. 220, London, 1997, pp. 375-390.
2. T. Zhou and K. Shimada: An Angle-Based Approach to Two-Dimensional Mesh Smoothing. In: Proc., 9th Int'l Mesh. Roundtable, New Orleans, 2000, pp. 373-384.
3. S. Canann, M. Stephenson, and T. Blacker: Optismoothing: An optimization-driven approach to mesh smoothing. Finite Elements in Analysis and Design **13** (1993), 185-190.
4. T. Li, S. Wong, Y. Hon, C. Armstrong, and R. McKeag: Smoothing by optimisation for a quadrilateral mesh with invalid element. Finite Elements in Analysis and Design **34** (2000), 37-60.
5. N. Amenta, M. Bern, and D. Eppstein: Optimal point placement for mesh smoothing. In: Proc., 8th ACM-SIAM Symp. on Disc. Alg., New Orleans, 1997, pp. 528-537.
6. S. Canann, J. Tristano, and M. Staten: An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes. In: Proc., 7th Int'l Mesh. Roundtable, Dearborn, Mich., 1998, pp. 479-494.
7. J. Freitag, M. Jones, and P. Plassmann: a Parallel Algorithm for Mesh Smoothing. SIAM J. on Scientific Computing **20** (1999), 2023-2040.
8. P. Hansbo: Generalized Laplacian Smoothing of Unstructured Grids. Communications in Numerical Methods in Engineering **11** (1995), 455-464.
9. D. Field: Laplacian Smoothing and Delaunay Triangulations. Comm. in Applied Numerical Methods **4** (1988), 709-712.
10. H. Xu: An Optimization Approach for 2D Finite Element Mesh Smoothing, M. S. Thesis, Dept. of Comp. Sci., Univ. of Ala. in Huntsville, Huntsville, 2003.
11. J. Nocedal and S. Wright: Numerical Optimization. Springer-Verlag, New York, 1999.