A COMPARATIVE STUDY OF MESH SMOOTHING METHODS WITH FLIPPING IN 2D AND 3D

By

TIKESHWAR PRASAD

A thesis submitted to the Graduate School - Camden

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Dr. Suneeta Ramaswami

And approved by

Dr. Suneeta Ramaswami

Dr. Sunil Shende

Dr. Jean-Camille Birget

Camden, New Jersey October 2018

THESIS ABSTRACT

A Comparative Study of Mesh Smoothing methods with Flipping in 2D and 3D

by TIKESHWAR PRASAD

Thesis Director:

Dr. Suneeta Ramaswami

Abstract: Mesh smoothing is the process of relocating mesh vertices with the goal of improving various quality metrics such as minimum angle, maximum angle and aspect ratio. This thesis presents a study of computationally efficient smoothing methods applied to triangle and tetrahedral meshes. It has been known in the meshing literature that mesh smoothing is more efficient when vertex relocation is accompanied by the edge flipping operation in order to maintain Delaunay properties. While flipping in 2D is simple and proven to be effective, flipping in 3D does not give such guarantees. In this comparative study, we implement four algorithms for triangular and tetrahedral meshing with immediate flipping. While immediate flipping has been mentioned in the mesh smoothing literature, it lacks experimental data, particularly for tetrahedral meshes. We compare these four methods in terms of speed, quality and grading of the mesh.

List of Figures

2.1.	Delaunay triangulation and empty circumcircle	3
2.2.	Possible triangulations in points in non general position	4
2.3.	Bowyer Watson Algorithm	5
2.4.	Two possible triangulations of four points	5
2.5.	Lawson Flip Algorithm	6
2.6.	1-4 flip	7
2.7.	2-3 flip	7
2.8.	4-4 flip	8
2.9.	Moving v to v_{new}	12
2.10.	. Sliver	13
3.1.	Feasible region in local Patch [2]	14
3.2.	Half-edge data structure	17
3.3.	Triangle and Tetrahedron with positive orientation	18
4.1.	Smoothing result on 2D mesh without flipping	22
4.2.	Smoothing result on 2D mesh with flipping	24
4.3.	Smoothing results on 3D mesh without flipping	26
4.4.	Smoothing results on 3D mesh with flipping	28
4.5.	3D models for sliver count	29

List of Tables

2.1.	Metrics table	12
2.2.	Ideal value table	12
4.1.	Qualitative comparison of smoothing methods in 2D \ldots	21
4.2.	Qualitative comparison of smoothing methods in 3D $\ldots \ldots \ldots$	25
4.3.	Sliver count without flipping	29
4.4.	Sliver count with flipping	29

Table of Contents

Abstract					
List of Figures					
\mathbf{List}	of	Fables			
1. I	ntr	duction $\dots \dots \dots$			
2 . E	Bac	ground			
2	2.1.	Delaunay Triangulation			
		2.1.1. Bowyer-Watson Algorithm			
		2.1.2. Lawson flip Algorithm			
2	2.2.	Smoothing Algorithms			
		2.2.1. Laplacian Smoothing			
		2.2.2. Smart Laplacian Smoothing			
		2.2.3. Angle Based Smoothing			
		2.2.4. Centroidal Voronoi Tessellation (CVT) Based Smoothing \ldots 10			
		2.2.5. Optimal Delaunay Triangulation (ODT) Based Smoothing 11			
		2.2.6. Aspect Ratio Based smoothing 11			
2	2.3.	Quality metrics			
2	2.4.	Slivers			
3. S	Smo	othing Methodology			
3	8.1.	Framework			
3	8.2.	$Implementation \dots \dots$			
		3.2.1. 2D mesh data structure $\dots \dots \dots$			

3.2.2. 3D mesh data structure \ldots	17
3.2.3. Robustness	18
4. Results and Discussions	20
4.1. Smoothing on 2D mesh	20
4.2. Smoothing on 3D mesh	24
4.3. Effect on Slivers	28
5. Conclusion and Future Work	31
5.1. 2D methods \ldots	31
5.2. 3D methods \ldots	31
5.3. Future work	32
References	33

Chapter 1

Introduction

Meshing is the process of discretization of a polygonal or polyhedral domain into simpler geometric elements like triangles, quadrilaterals, tetrahedra, hexahedra, etc; such that differential equations can be solved for simpler elements to model some real world problems for the entire geometric domain. These meshes are extensively used in FEA (Finite Element Analysis), FVM (Finite Volume Methods) and computer graphics. The accuracy of the results depends on the quality of the meshes particularly for FEA and FVM problems. A good quality mesh has a high number of good quality elements in the area of interest and fewer elements in other areas, which is important for grading. Grading can be defined as the variation of element size and count from different parts of the mesh. For example, good grading allows a large number of elements near a region of interest which varies smoothly to a small number of elements in other regions of the mesh. Acceptable quality depends on the nature of the problem, but elements with very small or very large interior angles are not considered to be of good quality.

The art and science of generating meshes is known as grid (or mesh) generation. Meshes can be differentiated into structured, unstructured and hybrid grids. Structured grids have regular and symmetrical connectivity and are easier to code, but they are not able to adapt to complicated domains. Unstructured grids have irregular connectivity and they are somewhat harder to code, but they can better adapt to complicated domains. Hybrid grids contain a mixture of structured and unstructured grids. Structured and hybrid grids are out of the scope of this thesis and will not be discussed further.

Among unstructured grid generation methods, Delaunay based methods have been

of special interest to researchers due to their nice mathematical properties and guarantees. Algorithms for Delaunay triangulation have been developed for 2D and 3D domains, but the initial mesh is usually coarse and not useful for solvers. For this reason, the mesh is often refined by adding more nodes to it in the regions of interest. Adding more nodes can improve mesh quality, but then solvers take more time to solve. If higher quality elements are desired after the mesh refinement, the mesh is passed through smoothing algorithms. Smoothing is the process of relocating mesh vertices to new locations such that elements incident at that vertex have improved quality. This is done iteratively for every vertex till the desired mesh quality is reached or it cannot be improved further. Smoothing can also involve changing the connectivity of nodes and change in connectivity can improve quality dramatically but is slower than vertex relocation alone.

There are many methods for applying smoothing on triangular and tetrahedral meshes, which can be broadly categorized into Laplacian based methods and Optimization based methods. The Laplacian based methods involve taking the average of some metric of elements, whereas optimization based techniques try to improve the metric by solving an optimization problem framed to minimize or maximize the chosen quality metric. It is not guaranteed that the Laplacian based methods will improve mesh quality; however the Laplacian methods perform reasonably well in practice given its simplicity. The Laplacian methods are easily extendable to quadilateral and hexahedral meshes. On the other hand, optimization based smoothing techniques yield better results but are computationally more expensive than Laplacian methods.

This thesis presents a comparative study of smoothing algorithms for two-dimensional meshes and their direct three-dimensional counterparts. The remainder of the thesis is organized as follows. In Chapter 2, we present the construction methods of Delaunay triangulations in 2D and 3D and discuss various smoothing methods and mesh quality metrics. In Chapter 3, we present our smoothing framework with our mesh data structure, implementation details and challenges. Chapter 4 gives an analysis of the effectiveness of smoothing methods on 2D and 3D meshes, followed by Chapter 5 on conclusion and future work.

Chapter 2

Background

2.1 Delaunay Triangulation

Delaunay triangulation is a method of triangulation in which every simplex satisfies the Delaunay property. In 2D triangulation, if every triangle has an empty circumcircle, then it is a Delaunay triangulation. Similarly in 3D, if every tetrahedron has empty circumsphere, then it is Delaunay. The empty circumcircle condition requires that there should not be any point inside or on that circumcircle in 2D or circumsphere in 3D except the points of the simplex itself as shown in Figure 2.1.



Figure 2.1: Delaunay triangulation and empty circumcircle

For a point set in general position, there always exists a unique Delaunay triangulation. In 2D, points are in general position when no 4 points lie on the same circle. If they lie on same circle, then there is more than one way to connect those 4 points to form a Delaunay triangulation as shown in Figure 2.2. To resolve this ambiguity, perturbation techniques are used.

There are many ways to compute the Delaunay triangulation of a point set. Two notable methods are incremental construction and divide-and-conquer. We will discuss



Figure 2.2: Possible triangulations in points in non general position

incremental construction here as that is easier to understand and code. There are two methods for incremental construction.

2.1.1 Bowyer-Watson Algorithm

The Bowyer-Watson algorithm [1, 13] constructs the Delaunay triangulation incrementally by inserting the points one at a time. To insert point p, first all the triangles are collected whose circumcicles contain p. The set of triangles are deleted to create a cavity. This cavity is re-triangulated with the p as shown in Figure 2.3. The algorithm is easily extensible to higher dimensions. In 3D, all tetrahedra are deleted whose circumspheres contain p forming a cavity. This cavity is then re-tetrahedralized to update the Delaunay tetrahedralization.

It is fairly easy to implement but is prone to robustness issues if robust predicates are not used. If points are not in general position, then Bowyer-Watson algorithm can lead to invalid triangulation due to floating point round-off errors. This can lead to catastrophic program crashes and failures.

2.1.2 Lawson flip Algorithm

The Lawson Flip algorithm uses the flip operation to construct the Delaunay triangulation. In case of 4 points in convex position, there are two possible triangulations,



(a) all triangles whose circumcircles contain the (b) updated triangulation with new point new point are deleted

Figure 2.3: Bowyer Watson Algorithm

but in general only one will be Delaunay; see Figure 2.4. Non-Delaunay configuration of 4 points can be converted to Delauanay by flipping the edge. In 2D, one can start with some arbitrary triangulation and then keep on flipping non-Delaunay edges until all edges are Delaunay. In incremental construction, a point p is inserted in following steps:



(a) Non-Delaunay Configuration circumcircle of $\triangle ABC$ contains D

(b) Delaunay Configuration circumcircle of $\triangle ABD$ does not contain C

Figure 2.4: Two possible triangulations of four points

- 1. First the triangle is found which contains the point p.
- 2. Each edge of that triangle is connected to the point to form 3 new triangles. The

old triangle is deleted.

3. Apply flipping to all the non-Delaunay edges starting from the new triangles and proceeding further with each new triangle until no non-Delaunay edge is left.

Theoretically, it can take n^2 flips to make an arbitrary triangulation of n points into Delaunay triangulation, but in practice, it is much less. See Figure 2.5 for flips in action.



edges

Figure 2.5: Lawson Flip Algorithm

This flipping [9] technique is quite useful in smoothing. After relocating every vertex to a new position, flipping is used to restore the Delaunay property.

Though the core of algorithm remains same in 3D, flipping is not as straight forward in 3D as in 2D. In 3D, edges and faces both can be flipped. Unlike in 2D, not every non-Delaunay configuration of edge/faces can be flipped. In 3D, to construct Delaunay tetrahedralization, we have two major types of flip operations[8]. A m-n flip operation means that m tetrahedra are replaced by n tetrahedra.

- 1. 1-4 flip : This is equivalent to inserting point in a tetrahedron and splitting into 4 tetrahedron.
- 2. 4-1 flip : This is equivalent to removing a point from the tetrahedral mesh. If exactly 4 tetrahedra are incident at a vertex, then they can be replaced by 1 tetrahedron thus deleting the vertex.



Figure 2.6: 1-4 flip

Here 3 tetrahedra are combined to create 2 tetrahedra.

- 2-3 flip : A face is flipped only when it is part of 2 tetrahedra and all points of 2 tetrahedra lie on the convex hull formed by the vertices of tetrahedra. So, 2 tetrahedra are flipped to give 3 tetrahedra.
- 4. 3-2 flip : A edge is flipped only when it is part of 3 tetrahedra and all vertices of 3 tetrahedra lie on the convex hull. Here 3 tetrahedra are combined to create 2 tetrahedra.



Figure 2.7: 2-3 flip

There are other flips as well, such as 2-2 flip and 4-4 flip, which are useful in degenerate cases. If 4 tetrahedra sharing an edge have degenerate points and 4 points are planar, then there are two possible configurations as shown in Figure 2.8 and one of better configurations can be chosen. If planar faces have exactly 2 tetrahedra on one side of the face, then we have 2-2 flip configuration which occurs only at boundary faces.



Figure 2.8: 4-4 flip

For performing flips after relocation of vertex to new point, flips are useful in 3D similar to 2D. Delaunay by flips has one major advantage over Bowyer-Watson that triangulation is always valid despite floating point errors. But in practice, flipping is slightly slower than former method.

2.2 Smoothing Algorithms

Laplacian smoothing is one of the most popular smoothing schemes. It comprises of those algorithms [5] that relocate the vertex to the weighted average of all its incident vertices. This can lead to inverted or invalid elements, so the extra check of mesh validity of is performed. However, it is still faster than other more sophisticated methods.

$$x^* = \frac{1}{k} \sum_{x_j \in \Omega_i, x_j \neq x_i} x_j \tag{2.1}$$

where x^* is the new location, x_i is the vertex in consideration and and Ω_i is the star of the vertex. *Star* of a vertex v is collection of all triangles in 2D or tetrahedra in 3D incident on v. The polygon created by the vertices in star is known as the *StarPolygon*. In practice, smoothing is an iterative method and is performed until certain threshold value of mesh quality is reached or a certain number of iterations are done. There are many methods to perform smoothing. In the following sections, we describe some of the commonly used smoothing methods in 2D.

2.2.1 Laplacian Smoothing

In Laplacian smoothing, a vertex v is simply relocated to the centroid of the vertices incident on v. This centroid point can be treated as the solution of a spring torsion system where each vertex in the star is fixed and applying force on v via a spring with constant K connected to it. Energy in such a system is

$$E = \frac{K}{2} \sum_{j=1}^{k} |v_j|^2 \tag{2.2}$$

where k is the number of vertices in the star of v. K is the spring constant, v_j is the vector from v to every vertex in the star. The solution of system is the centroid of all the vertices in the star.

It is simple and fast, however there is no guarantee for quality improvement and it might produce inverted elements. It has been observed that it can give good results if number of iterations are less.

2.2.2 Smart Laplacian Smoothing

The Smart Laplacian method addresses the shortcomings of the Laplacian method. The relocation strategy is the same but before relocating any vertex, it is passed through tests to see if it improves the quality of neighborhood simplices. The vertex's current position is not changed if there is no improvement. Despite the extra check for quality evaluation, this method is still inexpensive and avoids inverted elements [6].

2.2.3 Angle Based Smoothing

Angle based smoothing was developed by Zhou and Shimada [15]. This method can also be considered as solution to spring torsion system similar to the Laplacian method. The difference is that instead of distances, here angles made at v are considered. The energy in such a system is given by

$$E = \frac{K}{2} \sum_{j=1}^{2k} \theta_j^2$$
 (2.3)

where k is the number of vertices of the polygon, K the spring constant, and θ 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. Finding an optimum location is based on heuristics and notably there are two methods of finding new location due to Zhou and Shimada [15] and Xu and Newman [14].

This method is computationally more expensive than the Laplacian, but offers better results and avoids creating inverted elements without extra checks. We have not implemented this method for our study as there is no direct counterpart in 3D.

2.2.4 Centroidal Voronoi Tessellation (CVT) Based Smoothing

The Centroidal Voronoi tessellation [3] for a given point set is the collection of Voronoi cells such that the generating point of each cell is also its centroid. CVT is applied in image analysis, flow control, dimension reduction etc. CVT can be applied through LLoyd [10] iterations to the initial Delaunay triangulation to improve the mesh quality. In each Lloyd iteration, Voronoi regions of each point p is calculated and p is relocated to centoid of the Voronoi region. This is similar to k-means algorithm with the difference that k-means operates on points and Lloyd algorithm operates on continuous geometric Euclidean space.

From the implementation point of view, Lloyd iteration is slow to converge. Hence, researchers have come up with more deterministic ways to compute centroids. One of the easier ways is Weighted Centroid of Circumcenters which is shown below

$$x^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| C_j \tag{2.4}$$

where x^* is the new location, $|\Omega_i|$ is the total area of all the elements in the star of the point and T_j is *j*th triangle of star, |.| denotes the area in 2 dimension and C_j is the circumcenter of *j*th triangle. There are other methods based on CVT, but Lloyd iterations are well-suited for mesh smoothing despite its slow convergence.

2.2.5 Optimal Delaunay Triangulation (ODT) Based Smoothing

Chen and Xu [2] studied Delaunay triangulation for minimizing the linear interpolation error (measured in L^p -norm) for a given function. The Delaunay triangulation can then be characterized as an optimal triangulation that minimizes the interpolation error for the isotropic function among all the triangulations with a given set of vertices.

They devised a smoothing method based on this strategy which aims to distribute the edge lengths in the star of a vertex based on the function to be approximated. For a mesh with elements of roughly the same size also known as uniform mesh, the function can be chosen as $f(x) = |x|^2$. Then the new location can be computed as shown below.

$$v^{*} = -\frac{1}{2|\Omega_{i}|} \sum_{T_{j} \in \Omega_{i}} (\nabla|T_{j}(v)| \sum_{v_{k} \in T_{j}, v_{k} \neq v_{i}} |v_{k}|^{2})$$
(2.5)

In the above equation v^* is the new location, $|\Omega_i|$ is the area of all elements in the star of the vertex v, $\nabla |T_j(v)|$ is the vector towards v perpendicular to the edge opposite to v, $|v_k|^2$ is the square magnitude of the position vector of v_k and T_j is *j*th triangle incident on v.

2.2.6 Aspect Ratio Based smoothing

Aspect ratio based smoothing [7] method is a Laplacian based method which aims to improve mesh quality by targeting aspect ratios of the triangles incident at a vertex. The idea is that for each triangle T incident on v, we move v along the circumcircle of T such that the perpendicular distance of v from the edge opposite to v is maximized. We take the average of new locations calculated with respect to all triangles in the star of v and call it v* which is the final new location of v.

In the Figure 2.9, v is moved along circumcircle of triangle to v_{new} . The algorithm only moves the vertices if the ratio of lengths of longer to shorter edges incident at a vertex is greater than 1.5. We have extended this algorithm to 3D, where we try to maximize the distance of v from the face opposite to v by moving along the circumsphere.



Figure 2.9: Moving v to v_{new}

2.3 Quality metrics

Quality of an element of a mesh is measured by its suitability and accuracy in the simulation. Generally, very small or very large angles between edges or faces can impact accuracy and speed, so angles closer to equilateral elements are desired. Various quality metrics are used in the meshing community. Below are the most commonly used.

Table 2.1: Metrics table

Metric	Triangle	Tetrahedron
Aspect Ratio	$\frac{R}{2\sqrt{3}L}$	$\frac{R}{2\sqrt{6}L}$
Radius Ratio	$\frac{R}{2r}$	$\frac{R}{3r}$
Min Angle	min of 3 internal angles	min of 6 dihedral angles

R is radius of circumcircle in 2D or circumsphere in 3D. Similary r is inradius and L is the length of the shortest edge of the element. *Dihedral* angle is defined as angle between two planar faces. A tetrahedron has six pairs of dihedral angles between its four faces. Ideal values for the metrics are shown in Table 2.2

Metric	Triangle	Tetrahedron
Aspect Ratio	1.0	1.0
Radius Ratio	1.0	1.0
Min Angle (deg)	60	70.528

Table 2.2: Ideal value table

It has been shown that these metrics are related to each other and a bound on one metric implies a bound on the other metric. Aspect ratio or radius ratio is primarily used in refinement stage in 2D and 3D. Aspect ratio is used in refinement stage to split tetrahedra with poor aspect ratio values. However, aspect ratios are not helpful in identifying slivers. Slivers are very flat tetrahedra whose aspect ratios are as good as those of good quality elements. Radius ratio is more useful in identifying slivers. Slivers are discussed in some more depth in next section. There are other quality metrics which are variation of aspect ratio and radius ratio.

2.4 Slivers

In the context of mesh generation a sliver is a bad element which is flat with dihedral angles close to 0° or 180° . In Figure 2.10, all four points of tetrahedron are almost coplanar.



Figure 2.10: Sliver

There are other types of elements with poor shape like needle, wedge etc. but slivers constitute most of the bad elements in our meshes due to Delaunay condition. During refinement stage, elements with poor aspect ratios are eliminated by inserting more vertices in the mesh. However, slivers can have good aspect ratios close to the aspect ratios of nicely shaped elements, therefore they are not eliminated completely in mesh refinement stage. Sliver treatment is often done post refinement and smoothing as the last stage of mesh generation. We will discuss in detail the effect of the smoothing methods on slivers in the section of Chapter 4.

Chapter 3

Smoothing Methodology

Laplacian methods involve calculation of a new location which may lie outside of the star of the vertex. In such a case, inverted elements are created. If the star of the vertex is convex, then the new location is bound to lie inside the star. In theory, smoothing algorithms are applied to convex region of the star of the vertex [2] as shown in Figure 3.1 and convex star can be calculated using linear programming. But this is an expensive operation considering the fact that we want our smoothing methods to be computationally efficient. So to avoid inverted elements, we simply avoid relocating to the new vertex if it leads to inverted elements in 2D and 3D.



Figure 3.1: Feasible region in local Patch [2]

Furthermore, we immediately restore Delaunay property of the new location which works well in 2D and is theoretically guaranteed. But in 3D, there are simple configurations which are not Delaunay and can not be made Delaunay even by flipping. In addition to this, slivers exist in Delaunay configurations. One can try to avoid creating new slivers by performing tests on each flip operation; however it is an additional computation cost and configurations may end up not being Delaunay. To demonstrate the effectiveness of delaunizing immediately, we compared it with smoothing without flipping.

3.1 Framework

In this section, we present the framework for vertex relocation and flipping.

```
Algorithm 1 smoothing(flipflag)
while i \leq iterations do
   for every inner vertex v \ do
     \Omega \leftarrow \texttt{star region of } v
     v \ast \leftarrow smoothing(v, \Omega)
     if flipflag is true then
        if invertedElements(v*) is false then
           v \leftarrow v *
           makeDelaunay(v*)
        end if
     else
        minq \leftarrow worstQualityElement(v, \Omega)
        ming * \leftarrow worstQualityElement(v*, \Omega)
        if minq \leq minq * then
           v \leftarrow v *
        end if
     end if
   end for
end while
```

```
Algorithm 2 makeDelaunay(v)
```

```
\begin{array}{l} queue \leftarrow all \; non-delaunay \; edges/faces \\ \textbf{while} \; queue \; \textbf{is not empty do} \\ e \leftarrow dequeueTop() \\ \textbf{if } checkIfNotDelaunay(e) \; \textbf{is true then} \\ \lambda \leftarrow delaunayFlip(e) \\ enqueue(\lambda) \\ \textbf{end if} \\ \textbf{end while} \end{array}
```

We observed that naively moving the vertex to the new location can sometimes worsen the mesh depending on the quality criterion being evaluated. We tested two approaches; first we move the vertex v to new location v* followed by flipping only when it does not form inverted elements. Second, we move the vertex v to new location v* only when it improves the minimum quality measure in the star region and does not form inverted elements, and it is not followed by flipping. We discuss the results in chapter 4 in details for above mentioned approaches. We developed the above smoothing framework after considering these factors.

3.2 Implementation

We downloaded surface meshes from the stanford repository (http://graphics.stanford. edu/data/3Dscanrep/) and grabcad (https://grabcad.com/). We generated volume mesh using Tetgen (http://wias-berlin.de/software/tetgen/) for generating 3D meshes and Gmsh (http://gmsh.info/) for generating 2D meshes. We implemented 2D and 3D Delaunay algorithm in C++ using both Lawson's flip and Bowyer-Watson algorithm. Lawson's flip in 2D and it's 3D counterpart are extensively used in topological transformation of the meshes. We first read the input file and fill our mesh datastructure, and then perform the smoothing and flipping on it.

We used 2.2 GHz machine with 8 GB RAM to generate our results. We used Paraview based on VTK (https://www.vtk.org/) geometric library for generating snapshots and graphs for distribution of aspect ratios and min angles.

3.2.1 2D mesh data structure

Our data structure for 2D meshes is based on half-Edge data structure. It allows for output-sensitive query time, and one frequent query was finding all the triangles incident on a vertex. Below is a representation of two triangles based on the half-edge data structure. Edges a, b, c are half-edges of triangle face f1. And edges d, e, f are half-edges of triangle face f2 with d as the half-edge twin of c. A half-edge is oriented such that the triangle it bounds lies to its left. *Triangle* class has pointers to its 3 half-edges. Each half-edge has a pointer to its twin half-edge and the origin vertex. Notice that f1 and f2 are counter clockwise orientated. Oriented elements are useful in detecting inverted elements.

We implemented a *Tessellation2D* class which contains arrays of pointers of *Vertex*,



Figure 3.2: Half-edge data structure

Edge and Triangle. Tessellation2D data structure is filled when any file format containing 2D mesh is read from STL, OFF, VTK and mesh files. STL, OFF, VTK are different file formats to store mesh data as collections of vertices, edges and faces. Moreover, we implemented Lawson's flip algorithm to generate the Delaunay mesh on given points and boundary edges. Each vertex is inserted in an incremental manner followed by edge recovery using only flips. So, with above mentioned methods, Tessellation2Ddata structure is filled and passed onto our smoothing framework.

3.2.2 3D mesh data structure

Our 3D mesh data structure is loosely based on half-face data structure. Similar to 2D, we maintain orientation of the elements. Tetra class contains pointers to its 4 vertices and 4 faces. Each *Face* class contains pointers to 3 edges and 3 vertices. Each *Edge* has pointers to its 2 end points vertices. There is no half-face or half-edge, because we maintain orientation by ordering of the vertices such that volume given by that order is positive.



Figure 3.3: Triangle and Tetrahedron with positive orientation

Similar to 2D, we implemented a *Tessellation3D* class which contains arrays of pointers of *Vertex*, *Edge*, *Triangle* and *Tetra*. *Tessellation3D* data structure is filled when any file format containing 2D mesh is read from vtk and mesh files. Stl and off files are not adequate enough to represent tetrahedral mesh. We used Tetgen [12] to generate 3D mesh for given input points and surface mesh. And we implemented 3D flips presented in section 2.1 to perform flipping in smoothing which were first described in detail in Joe's [8] flip algorithm.

3.2.3 Robustness

Delaunay based algorithms are quite sensitive to floating point errors. Despite correctness of algorithm and efficiency of data structure, it is very hard to scale to a large number of points. One of the common problems faced was inverted or overlapping elements which would later result to incorrect data causing the program to crash. So, for robustness, we used Jonathan Schewchuck's [11] code for robust predicates which is free for academic purposes. Schewchuck presents an adaptive technique, where the exact arithmetic is used only to determine the sign of the determinant and not its value, thus saving on the computational speed. Alternatively, one can use CGAL's exact arithmetic predicates. Delaunay's code needs only two or three predicates orient2D, orient3D, inCircle, inSphere so getting them right reduces floating point errors significantly. Both Schewchuck's and CGAL's predicated are tested and proven for their robust computations. Another issue was repetition of points or almost identical points in input files. Tolerance value of 10^{-8} is employed to distinguish between two points. While inserting vertex v in data structure, v is checked for closeness to its surrounding triangle or tetrahedral vertices, and if distance to any other vertex from v is less than our tolerance value, then v is not inserted. Precision arithmetic is extremely critical for tetrahedral meshes.

Chapter 4

Results and Discussions

In this chapter we will compare the results of our smoothing algorithms with each other with and without flipping in 2D and 3D. First we present a comparison of smoothing algorithms in terms of speed, quality and grading of the meshes on the scale of Poor, Medium, Good. Then we will show the results based on distribution of min angles and radius ratios. In figures 4.1 and 4.2, first on the left are mesh output, in the middle are distribution of min angles, on the right are distribution of radius ratios. For aspect ratios, all elements with aspect ratios ≥ 5 are characterized as bad elements due to deformed shapes. In all iterations, smoothing are performed on interior vertices. Handling of boundary vertices needs extra care and needs an extension to the algorithms [4] to accommodate them. Inner vertices can be moved freely inside the mesh whereas vertices on the boundary can only be moved along the boundary edge or face which requires extensions to current smoothing algorithms.

4.1 Smoothing on 2D mesh

Figures 4.1 and 4.2 show smoothing algorithms with and without flipping respectively. The square contains approximately 300 points and original figure 4.2a shows the unsmoothed Delaunay triangulation. The histogram in the middle shows the distribution of minimum internal angles of triangles. The histogram on the right shows the distribution of radius ratios. It's a bad triangulation to start with but effectiveness of the smoothing algorithms can be seen from mesh output.

Smoothing method	Ease of	Speed	Quality	Grading
	Implementation			
Smart Laplacian	Good	Good	Medium	Good
Optimal Delaunay	Medium	Good	Good	Poor
Triangulation based				
Aspect ratio based	Medium	Medium	Good	Poor
Average circumcenter	Good	Medium	Good	Good

Table 4.1: Qualitative comparison of smoothing methods in 2D

2D smoothing without flipping



(b) Smart laplacian



(e) Average circumcenter

Figure 4.1: Smoothing result on 2D mesh without flipping

Smart Laplacian has previously been reported to perform well despite its simplicity [5]. Average circumcenter also performs well. Both methods are able to move elements to good radius ratios. ODT did not improve the mesh quality much which is evident from the number of elements with radius radio \geq 5.

2D smoothing with flipping



(c) Optimal Delaunay triangulation



(e) Average circumcenter

Figure 4.2: Smoothing result on 2D mesh with flipping

Mesh improvement is greater with flipping as flipping restores Delaunay property, and Delaunay triangulation maximizes the minimum angle. ODT performs the best in terms of overall improvement and effectiveness of ODT was observed on other meshes too. Aspect ratio comes close to ODT followed by average circumcenter.

4.2 Smoothing on 3D mesh

Figure 4.4 shows smoothing algorithms with flipping in 3D. The mesh shown is a crosssection of cylinder with approximately 1000 vertices, most of them are on the surface. Original figure 4.4a shows the unsmoothed Delaunay tetrahedralization. The histogram in the middle shows the distribution of minimum internal dihedral angles and the one on the right shows the distribution of radius ratios. It's a refined mesh to start with

Smoothing method	Ease of	Speed	Quality	Grading
	Implementation			
Smart Laplacian	Good	Good	Poor	Poor
Optimal Delaunay	Medium	Medium	Good	Poor
Triangulation based				
Aspect ratio based	Medium	Medium	Good	Poor
Average circumcenter	Good	Good	Good	Good

but smoothing nevertheless improves the overall quality of the mesh.

Table 4.2: Qualitative comparison of smoothing methods in 3D

3D smoothing without flipping



(b) Smart Laplacian



Figure 4.3: Smoothing results on 3D mesh without flipping

We observe that Laplacian does not perform well compared to other methods. Other methods are able to produce more elements with better minimum dihedral angles but they may still perform poorly when it comes to slivers. For slivers, the radius ratio distribution is a better indicator. The number of elements with a spect ratio ≥ 5 contains slivers too.



3D smoothing with flipping

(c) Optimal Delaunay triangulation



(e) Average circumcenter

Figure 4.4: Smoothing results on 3D mesh with flipping

In 3D, analysis is not straight forward as in 2D. ODT improves overall aesthetics of the mesh but it can worsen sliver counts as discussed in the next section. Average circumcenter consistently performs the best, and is able to preserve grading, improves the sliver situation, and is one of simpler and faster smoothing methods.

4.3 Effect on Slivers

A Sliver is a very flat tetrahedron such that the configuration of its vertices can be Delaunay. The minimum dihedral angle in 3D Delaunay is not guaranteed, therefore, there is no known method which produces sliver free tetrahedralization for all kinds of vertex sets. Generally sliver treatment is taken as a post process step to improve the mesh. Smoothing in general can help reduce the number of slivers but some may aggravate the sliver situation. We tested the effect of our smoothing methods on the mesh in the context of sliver counts. We measured the number of slivers before and after out smoothing methods on the models shown below.



Figure 4.5: 3D models for sliver count

Smoothing method	liver	hand	yakima
Original	10	735	75
Smart Laplacian	6	532	61
Optimal Delaunay Triangulation based	7	600	69
Aspect ratio based	6	504	58
Average circumcenter	5	494	58

Table 4.3: Sliver count without flipping

Smoothing method	liver	hand	yakima
Original	10	735	75
Smart Laplacian	1933	2505	37
Optimal Delaunay Triangulation based	690	1856	116
Aspect ratio based	330	1308	85
Average circumcenter	29	278	41

Table 4.4: Sliver count with flipping

We generated the mesh using Tetgen [12] without any optimization switches on. These meshes were of good quality to start with but still have tetrahedral elements with very small or very large dihedral angle. To capture such elements, we counted the elements whose radius ratio is greater than 10.0. Most of the slivers were captured along with some bad elements which may not qualify as slivers. Tetrahedral mesh improvement remains an active area of research in terms of providing guarantees. Clearly our results show the heuristic nature of smoothing but average circumcenter method outperformed all other methods with and without flipping. Our 3D counterpart of Aspect Ratio [7] based smoothing improves upon Optimal Delaunay Triangulation [2] based smoothing.

Chapter 5

Conclusion and Future Work

This thesis presents a comparative study of smoothing methods that are not computationally expensive and improve the original mesh effectively. The Laplacian based methods can be used as a precursor to more computationally expensive methods. As the Laplacian is based on heuristics, there is no provable guarantee that for every shape and size of model, they are effective. In practice, however they perform well and are faster.

5.1 2D methods

The Results section showed that flipping right after vertex relocation produces better quality meshes. For 2D meshes, flipping works very well as it has been shown that Delaunay meshes maximize the minimum angle and flipping restores the Delaunay property of the vertex in consideration. Of the four methods, ODT produced the mesh with the most triangles that are close to equilateral. On the other hand, ODT is the slowest, taking more to converge to the given epsilon displacement.

Aspect ratio based method is comparable to ODT and slightly faster than ODT. Average circumcenter method is able to maintain some grading of the meshes and given some mesh density function, the Laplacian and average circumcenter methods can be easily modified as per mesh density.

5.2 3D methods

For 3D meshes, slivers are a big problem and can not be avoided either in refining or in smoothing stages. Smoothing can help reduce the number of bad elements as seen in the case of average circumcenter for 3D meshes. Nevertheless, a Postprocessing of slivers is advised after performing smoothing stage. Naive implementation of flipping after vertex relocation can aggravate the sliver situation. This is evident from the fact that in 3D, flipping is not guaranteed to restore the Delaunay property of the star polyhedron of the vertex, which is why in 3D, we can have arrangement of tetrahedron which are not Delaunay. Also, slivers are formed even in Delaunay based configurations of vertices. While flipping, an extra check is needed to make sure that current configuration is free from slivers, which adds to the cost of the methods.

Overall, ODT performs well for 2D with immediate flipping and average circumcenter for 3D meshes.

5.3 Future work

We have implemented a direct extension of aspect ratio based smoothing2.2 and compared other cost effective smoothing methods without any special handling of the vertices on or near the boundary. In future, we would like to extend these methods to handle the boundary. Especially for tetrahedral meshes, handling of boundaries is challenging even for simple smoothing methods. Also, we would like to see how these methods can be extended to smoothing of quadilateral and hexahedral meshes.

References

- A. Bowyer. Computing Dirichlet tessellations. The Computer Journal, 24(2):162– 166, January 1981.
- [2] Long Chen and Jinchao Xu. Optimal Delaunay triangulations. Journal of Computational Mathematics, 22(2):299–308, 2004.
- [3] Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. SIAM J. Numerical Analysis, 44:102– 119, 2006.
- [4] Hale Erten, Alper ngr, and Chunchun Zhao. Mesh smoothing algorithms for complex geometric domains. In *Proceedings of the 18th International Meshing Roundtable*, pages 175–193. Springer Berlin Heidelberg, 2009.
- [5] David A. Field. Laplacian smoothing and delaunay triangulations. Communications in Applied Numerical Methods, 4(6):709-712, 1988.
- [6] Lori A. Freitag. On combining laplacian and optimization-based mesh smoothing techniques. pages 37–43, 1997.
- [7] Xuan Huang and Dianna Xu. Aspect-ratio based triangular mesh smoothing. In ACM SIGGRAPH 2017 Posters, SIGGRAPH '17, pages 68:1–68:2, New York, NY, USA, 2017. ACM.
- [8] Barry Joe. Construction of three-dimensional delaunay triangulations using local transformations. Comput. Aided Geom. Des., 8(2):123-142, May 1991.
- [9] C. L. Lawson. Mathematical Software III; Software for C1 surface interpolation. pages 161–194, 1977.
- [10] S. Lloyd. Least squares quantization in pcm. IEEE Trans. Inf. Theor., 28(2):129– 137, September 2006.
- [11] Jonathan Richard Shewchuk. Robust Adaptive Floating-Point Geometric Predicates. In Proceedings of the Twelfth Annual Symposium on Computational Geometry, pages 141–150. Association for Computing Machinery, May 1996.
- [12] Hang Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw., 41(2):11:1–11:36, February 2015.
- [13] D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, January 1981.
- [14] Hongtao Xu and Timothy S. Newman. 2d fe quad mesh smoothing via angle-based optimization. pages 9–16, 2005.

[15] Tian Zhou and Kenji Shimada. An angle-based approach to two-dimensional mesh smoothing. pages 373–384, 2000.