Tetrahedral Meshes in Biomedical Applications: Generation, Boundary Recovery and Quality Enhancements

Hamid R. Ghadyani

A dissertation submitted to the faculty of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in Mechanical Engineering

March 2009

Professor John M. Sullivan, Jr. (Advisor)

Professor Allen H. Hoffman (Committee Member)

Professor Brian J. Savilonis (Committee Member)

Professor Subhadra Srinivasan (Committee Member)

Professor Mark Richman (Graduate Committee Representative)

Acknowledgements

It would be difficult to acknowledge those people who have contributed to this work without either making an omission or exceeding the proper page limits.

First, I would like to thank my advisor, John M. Sullivan, for all his guidance and support. Much of this work was developed in countless discussions with John. Many of the insights for this work are his, while the deficiencies are mine only. He has been an outstanding mentor and supporter in my academic life and beyond. Without his famous stuffed turkey at Thanksgiving nights, enduring cold and long winters of Massachusetts would have been difficult. I could not have completed my education without the constant support and encouragement of my family: my father and mother, my two brothers and my beautiful sister. Also I would like to thank Dr. Allen Hoffman, Dr. Brian Savilonis, Dr Subhadra Srinivisa and Dr. Richman for their service on my dissertation committee. I am grateful for all the help that has always been available to me with a smile by Barbara Edilberti, Barbara Furman, Pamela St. Louis, the kindest and nicest people of the Mechanical Engineering Department. I would like to express my gratitude to Sia Najafi for *all* help and support he provided for a young international student in a new country. I was lucky to have Murali Murugavel, Praveen Kulkarni , James Zhang, Wei Huang and Udo Benz as my colleagues, they made my stay at WPI a great experience.

Also partial funding for this work was provided by National Institute of Health award P01CA080139-08.

If this thesis is defended, I will surely celebrate with BJ.J., B.K., N.K., K.S. and N.Kz.

Tetrahedral Meshes: Generation, Boundary Recovery and Quality Enhancements

Hamid R Ghadyani

March 30, 2009

Abstract

Mesh generation is a fundamental precursor to finite element implementations for solution of partial differential equations in engineering and science. This dissertation advances the field in three distinct but coupled areas. A robust and fast three dimensional mesh generator for arbitrarily shaped geometries was developed. It deploys nodes throughout the domain based upon user-specified mesh density requirements. The system is integer and pixel based which eliminates round off errors, substantial memory requirements and cpu intensive calculations. Linked, but fully detachable, to the mesh generation system is a physical boundary recovery routine. Frequently, the original boundary topology is required for specific boundary condition applications or multiple material constraints. Historically, this boundary preservation was not available. An algorithm was developed, refined and optimized that recovers the original boundaries, internal and external, with fidelity. Finally, a node repositioning algorithm was developed that maximizes the minimum solid angle of tetrahedral meshes. The highly coveted 2D Delaunay property that maximizes the minimum interior angle of a triangle mesh does not extend to its 3D counterpart, to maximize the minimum solid angle of a tetrahedron mesh. As a consequence, 3D Delaunay created meshes have unacceptable sliver tetrahedral elements albeit composed of 4 high quality triangle sides. These compromised elements are virtually unavoidable and can foil an otherwise intact mesh. The numerical optimization routine developed takes any preexisting tetrahedral mesh and repositions the nodes without changing

the mesh topology so that the minimum solid angle of the tetrahedrons is maximized. The overall quality enhancement of the volume mesh might be small, depending upon the initial mesh. However, highly distorted elements that create ill-conditioned global matrices and foil a finite element solver are enhanced significantly.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Meshes are an indispensable tool in a wide variety of fields ranging from computer graphics, CAD systems to numerical simulations and partial differential equations. The work presented in subsequent chapters was motivated by increasing Biomedical Engineering applications for Finite Element Method (FEM) and the need for robust and reliable mesh generators. The focus of the effort is restricted to triangular (2D) and tetrahedral (3D) meshes. These mesh types are dominant element configurations among the various types of meshes. This chapter briefly illuminates some mesh generation complications that have plagued the field for years and their resolutions based on the work presented herein.

1.1 Meshing and its challenges

In order to solve a partial differential equation (PDE) using FEM, the domain of problem needs to be discretized into smaller regions called elements. Because elements

have simple shapes relative to the original object, the behavior of a PDE can be approximated across the simplifed geometry. A FEM system of equations is a linear combination of these approximations from each element assembled into a large global matrix. The discretization process or mesh generation can pose more problems than the rest of numerical simulation process [94]. As a consequence, mesh generation strategies have been researched extensively over the decades of the FEM solution methodology.

One very common approach to mesh generation is to first deploy a set of vertices/nodes within the physical domain and then form a topology (or connectivity list) using the nodes. Within this area of topology formation, *Delaunay* triangulation (2D) or tetrahedralization (3D) is the dominant method because of its sound theoretical foundations. It is based on the idea of connecting nodes that share an edge in a Voronoi diagram. A set of given points along with line edges (for 2D) or *piecewise linear complex*[1] (for 3D) as the boundary definition is passed to Delaunay algorithms in order to create the mesh.

1.1.1 Delaunay Tessellation

A triangular mesh is called Delaunay if all the circumcircles[2] of the mesh triangles are empty. An empty circumcircle does not contain any vertex of the mesh. Figure 1.1 shows a set of points and its Delaunay triangulation as well as the empty circumcircles. The same can be applied for a set of tetrahedrons in 3D. They are Delaunay if and only if all the circumspheres of the mesh are empty. In both cases the vertices can

[1] A set of vertices, segments and facets all of which are compromised of planar straight line graphs
[2] circumbscribed circle: a circle that passes through all vertices of a polygon

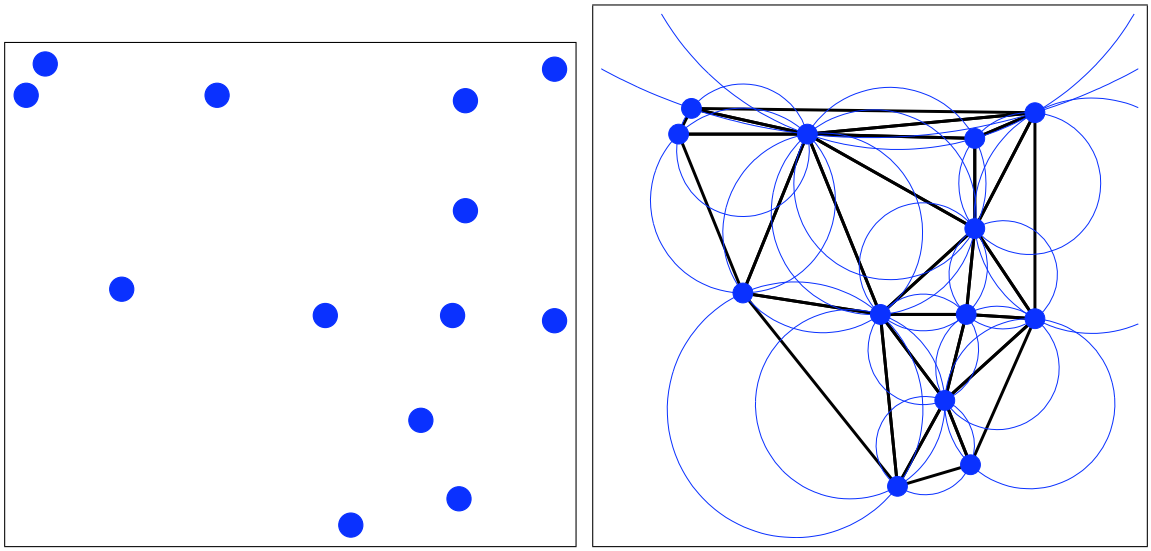


Figure 1.1: (a) Scattered nodes in 2D. (b) Delaunay triangulations of the nodes. None of the circles shown encloses a node, thus *empty* circles.

exist on circle or sphere.

1.1.2 Obstacles

In two dimensions, a Delaunay mesh maximizes the minimum angle found over the entire mesh. This is a very desirable property since small angles in a mesh have been shown to have negative effects on the outcome of the FEM solutions such as larger errors in numerical approximations and also impact on the performance of the simulations. These effects are discussed in Chapter 4. Unfortunately, this desirable property of a Delaunay mesh in two dimensions can not be extended to its 3D counterpart, i.e. to maximize the minimum solid angle of the mesh. Figure 1.2 shows an example in which a non-Delaunay formation produces a better and *round* (Figure1.3) shaped element. Other forms of compromised tetrahedrons can be generated by Delaunay

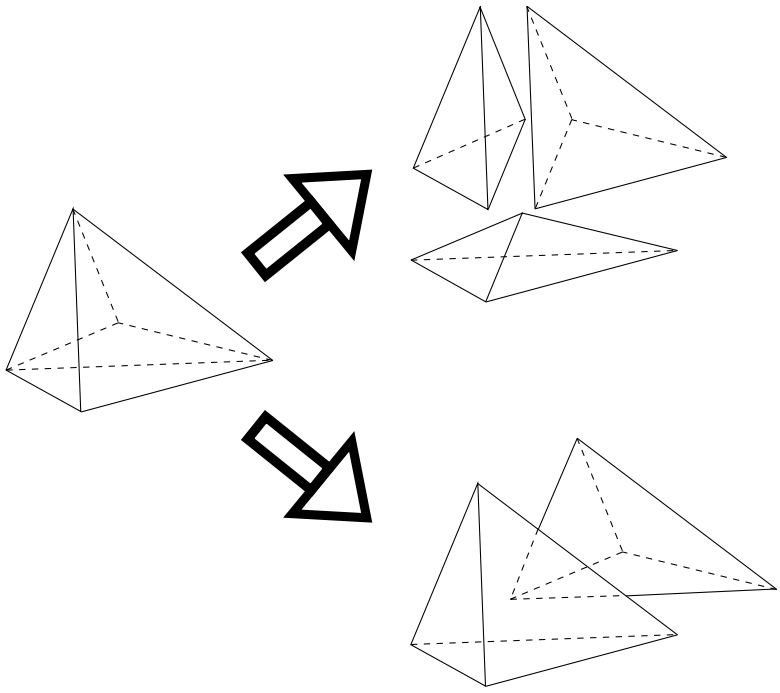


Figure 1.2: Delaunay tetrahedrons (top) have an almost flat element while a non-Delaunay mesh produces higher quality elements for this configuration (bottom).

method as shown in Figure 1.3.

A mesh generator is often required to create elements that are in compliance with the sizing requirements of the underlying governing equation rather than the geometry of the domain. This situation requires a spatially variable mesh resolution. To achieve better approximations in FEM or satisfy stability constraints, smaller element sizes are required in some regions, say, close to boundaries of domain while moving away from the boundary the size restrictions can be relaxed. One might be tempted to create a fine resolution mesh all over the domain. However, a variable or graded resolution mesh saves computer resources (because it deals with fewer elements) and reduces computational load and time required to solve the FEM problem.

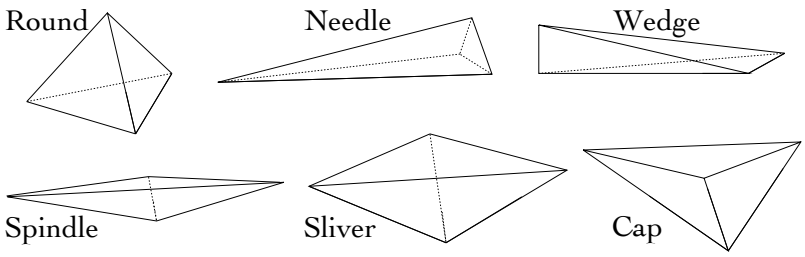
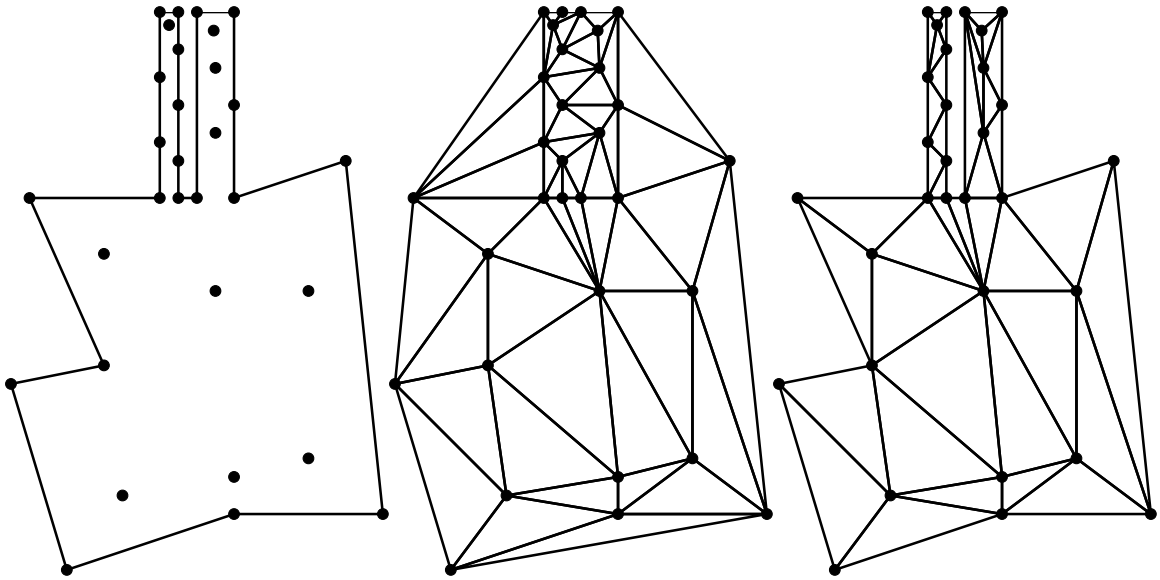


Figure 1.3: Different forms of compromised tetrahedral elements compared to the ideal "Round" tetrahedron.

Frequently, the original input surface (nodes and connectivity) is required to be part of the final volume mesh. For example when applying a flux across a surface or material change interface the boundary conditions might be phrased in terms of surface facets. A numerical formulation that maintains conservation of the physics requires a consistent mesh. That is, the mating faces from one element to another (or across a boundary interface) must match identically. Delaunay triangulation or tessellation guarantees a consistent mesh within the interior. However, it does not preserve the original boundary edges (2D) or faces (3D) in the final mesh. Figure 1.4 is a 2D example of this caveat in a non-*constrained* Delaunay. The mesh in the middle is missing some of boundary segments that delineate the domain of the problem. These segments can be recovered using a *constrained* Delaunay which does not split the segments as oppose to *conforming* Delaunay. However, an algorithm to ensure a constrained mesh in 3D has been sorely lacking.

After a mesh generator is done and the location of all the nodes are determined, one might be able to improve the quality of mesh by using some type of smoothing method or optimization of node locations based on a given objective function. Generally these methods do not alter the topology of the mesh rather they attempt to reposition the



(a) *Original boundary* (b) *Delaunay triangulation* (c) *Constrained Delaunay*

Figure 1.4: (a) A 2D boundary defined by line segments. (b) Delaunay triangulation of the domain. (c) *constrained* Delaunay triangulation preserving all input segments with fidelity.

nodes to improve the quality of the neighboring elements. These methods prove to be more useful in 3D than 2D since there are more possibilities to form undesirable elements[34].

Last but not the least concern is round off errors of floating-point calculations involved in all numerical methods. Result inaccuracies can exist for any procedure that operates on floating-point numbers but geometrical algorithms are especially vulnerable. A typical numerical algorithm properly framed will keep the round off inaccuracies within the noise of the system. However, a geometrical algorithm to determine if 3 straight lines intersect at the same location can fail due to these round off errors.

1.2 Dissertation advances

The goal of this research is to offer better solutions to the issues mentioned in 1.1.2. The next 3 chapters each deal with one dilemma previously mentioned in tetrahedral mesh generation:

1. Node deployment and *grading* mesh generation,
2. Boundary recovery and sliver removal,
3. Mesh smoothing and optimization.

Each chapter reviews the current state of the art and related work in literature and offers an extension to it. At the end of each chapter results and examples are provided to better demonstrate the algorithms.

Chapter 2 deals with a mesh generator that provides the user with a great deal of flexibility in sizing the mesh element. Grid generation and domain discretization for Finite Element applications are usually done based upon three general methods: *"node placement and connection", "domain decomposition"* and *"mesh-template mapping"* or a combination of these methods. In this chapter a method is proposed that combines all above methods by using a template matrix overlaid on domain of interest. The template overlay is used to guide node placement within the domain. The advantage of using a grid is substantial. It eliminates error-prone floating-point calculations and avoids expensive *exact geometric* computations. The proposed method can be considered a fusion between octree technique and offsetting/advancing front method without the extensive processing of working fronts.

In Chapter 3, I deal with element quality and boundary recovery problems. Numerous high-quality, volume mesh-generation systems exist. However, no strategy can address all geometry situations without some element qualities being compromised. Many 3D mesh generation algorithms are based on Delaunay tetrahedralization which frequently fails to preserve the input boundary surface topology. For biomedical applications, this surface preservation can be critical as they usually contain multiple material regions of interest coherently connected. In this chapter I present an algorithm as a *post-processing* method that optimizes local regions of compromised element quality and recovers the original boundary surface facets (triangles) regardless of the original mesh generation strategy. The algorithm carves out a small sub-volume in the vicinity of the missing boundary facet or compromised element, creating a cavity. If the task is to recover a surface boundary facet, a natural exit hole in the cavity will be present. This hole is patched with the missing boundary surface face first followed by other patches to seal the cavity. If the task was to improve a compromised region, then the cavity is already sealed. Every triangular facet of the cavity shell is classified as an active face and can be connected to another shell node creating a tetrahedron. In the process the base of the tetrahedron is removed from the active face list and potentially 3 new active faces are created. This methodology is the underpinnings of our LAST RESORT method. Each active face can be viewed as the trunk of a tree. An exhaustive breath and depth search will identify all possible tetrahedral combinations to uniquely fill the cavity. We have streamlined this recursive process reducing the time complexity by orders of magnitude. The original surface boundaries (internal and external) are fully restored and the quality of compromised regions improved.

In Chapter 4, I propose a method that optimizes the quality of any tetrahedral mesh regardless of element generation method. It extends the most sought 2D Delaunay property to 3D, i.e. maximizing the minimum solid angle. The algorithm identifies an interior node and an associated sub-volume of the mesh. It moves the interior node in a direction to maximize the minimum solid angle. The process of node and sub-volume identification is done iteratively until no further solid angle improvement is possible. The proposed method has the option of altering the boundary nodes of the mesh in a bid to maximize the optimization. If this option is chosen, it moves the boundary nodes only in a tangential direction, i.e. they simulate a sliding motion on the triangular facets of the original surface boundary. The results show the low quality or sliver elements improve dramatically which can make all the difference between a numerically sound mesh and one that foils a solver.

Chapter 2

Template Based Node Deployment

2.1 Background

The finite element method contributes significantly to advances associated with the coupling of engineering and the life sciences [46, 13, 45, 85, 118]. Numerous approaches have been proposed and developed for mesh generation (structured or unstructured) during past twenty five years. The techniques being widely used include advancing front approaches [38, 69, 50], Delaunay triangulations [90, 107, 40, 84], quad-tree techniques [79, 110, 73] and sphere packing [67, 102]. It is noteworthy to mention that these classifications consist of other sub-processes that can be shared by mentioned methods. Most important of which are node placement and connection; domain decomposition and mesh-template mapping [48].

In quad-tree methods, a cube grid containing the object to be discretized is recursively subdivided until a desired resolution is reached after which nonuniform elements

are created at the intersection of the partitioned cubes and the boundary[119]. However, this category does not preserve the input boundary. It uses the input boundary to define the limits of the domain. It can also require a remarkable number of intersection operations depending upon mesh density. Another variation of grid-based approaches is creating a fitted mesh of elements into the volume with elements added at the boundary to fill the gaps of regular grid [88]. This method, while being robust, can create low quality elements at the boundary due to difficulties in filling the gaps between the boundary and the grid[78].

Delaunay triangulation/tetrahedralization is used extensively in numerous mesh generation strategies for practical engineering applications[113, 102, 57, 40]. A criterion frequently associated with the 3D Delaunay method is the "empty sphere" which states that no circumsphere of any tetrahedron in the mesh should contain any nodes of the mesh. The work of Lawson[58] and Watson[111] initiated the popularity of this method although it was known for many years[23]. In most implementations of Delaunay triangulation, there is no guarantee that all original boundary segments will be present in the final mesh triangulation. It should be noted that Delaunay method provides an algorithm to form topology of the mesh using existing nodes and does not supply any measure to position the nodes. Therefore it has been combined with various techniques to generate the nodes. Work done by Weatherill[114] used the centroid of the tetrahedrons to place nodes. Other methods [24] extend the work of Chew[19] and Ruppert [86] and used centers of circumsphere to place new nodes which can create meshes with guaranteed bound on angles in the mesh.

A major issue in Delaunay method is related to preserving the original topology

that defined the domain of interest. This lack of conformity was addressed by multiple researchers. Weatherill[114] demonstrated how edge swapping can help recover some of the initial boundary segments. George et. al. [43, 9] extended this work to recover boundary edges and faces for a three-dimensional tetrahedral mesh. However, they inevitably introduced new nodes and facets that altered the original boundary surface. We recently presented [44] a new approach to the problem in which local topological transformations were performed to recover original boundary features. The method does not introduce new nodes to the mesh and restores the original surface topology with fidelity. If desired, this strategy can be used exhaustively to examine all possible formations to find an optimum solution.

Advancing front approach was first proposed by Lo[65, 66] and Lohner[68] and was later modified for the generation of quadrilaterals and extended into three-dimensions, surface meshing and other applications. In this method, the elements are built progressively inward from the triangulated facets of the boundary. For every existing facet the location of a new 4th node is calculated to form a quality element. As elements are formed new sets (layers) of triangulated facets are introduced. This front layer keeps advancing toward interior of the domain as the algorithm fills it with elements. Along similar lines the normal offset strategy by Johnston and Sullivan deployed interior nodes based upon well shaped interior layers[54]. These methods require multiple intersection and enclosure tests, especially when it comes to opposing layers approaching each other, to ensure the integrity of the final mesh.

The sphere/circle packing technique is a node placement tool in domain of interest which also utilizes an element generation method, such as Delaunay. Given a domain

geometry and a node-spacing function, spheres are packed on geometry simplexes such as vertices, edges and faces. The interior nodes will then be placed in the center of spheres [101]. Different variations of this method have evolved during past years. Bern et al. [7] and Eppstein[29] triangulated a n-vertex polygonal region (with holes) by packing circles of different radii so that the angle was bounded by $\frac{\pi}{2}$. Li et al. combined this method with advancing front strategies.

One of the challenges for mesh generation algorithms is robustness for multiple situations. Frequently, a cumbersome geometry can foil an otherwise solid algorithm. Large scale problems can be compromised by floating-point round-off operations which are a consequence of the numerical geometry calculations[49]. There are methods[92] proposed that guarantee this robustness for some geometrical predicate operations, however they are not inclusive of all geometries nor physics based constraints. Frequently, the element size requirements are imposed by the application physics or user specified constraints, rather than geometry considerations.

2.1.1 Proposed method

Generating a quality mesh that provides accurate numerical results is fundamental for FEM analysis. In this paper, we propose a robust 3D mesh generation strategy that creates quality meshes. Starting with a boundary that defines the problem domain, we overlay the entire volume with a uniform grid of voxels whose spacing is a function of the finest desired resolution which is frequently related to the minimum edge size in the boundary surface. Initially, one might fear the memory requirements of such a dense deployment. However, this grid is pixel or integer based. Further, each

storage voxel requires less than 1 Byte. Consequently, the uniform grid deployment requires significantly less memory than the final tetrahedral mesh. The voxel overlay is mainly used to guide node placement within our algorithm. The advantage of using a grid is that we do not have to deal with floating-point calculations and performing numerical geometry operations and tests. Further moving around the domain is simple and robust. Initial physical boundary nodes are identified in the overlaid grid and they are used as seeds to determine the location of other interior nodes. While the algorithm moves away from the boundary into the domain interior, it maintains the desired element size (spatially dependent) while placing new nodes.

2.2 Methodology

Determining the node locations to ensure appropriate resolution throughout the volume is an important step of mesh generation. In the process, two factors need to be addressed: desired mesh density at a given location and quality of the elements to be formed. Using a 3D template matrix, described in section 2.3, we streamline the process of node deployment in order to satisfy these requirements. The main steps of the algorithm can be defined as follows:

1. Read in the surface mesh that defines the volume of interest.

2. Create a template matrix enclosing the entire domain.

3. Identify the template matrix pixels corresponding to initial boundary nodes.

4. Based on density requirements, traverse pixels of template matrix and mark

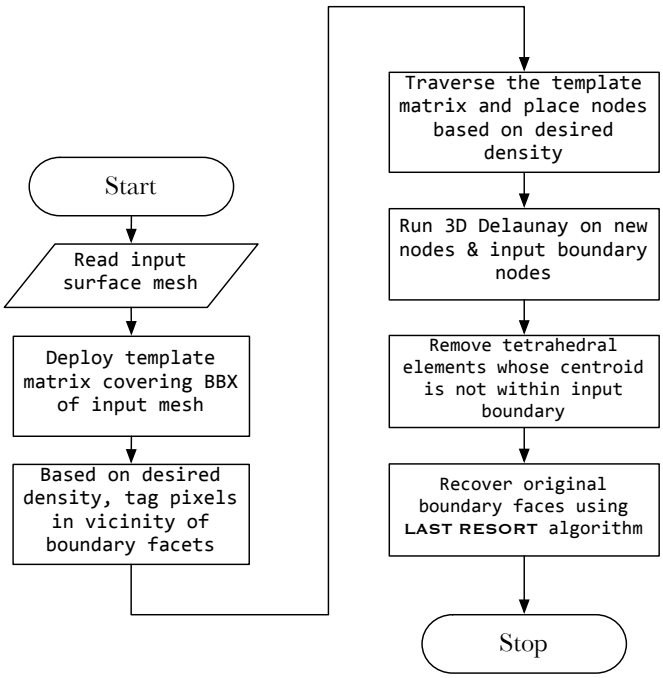


Figure 2.1: General flowchart of mesh generation method.

them as interior vertex locations.

5. Using set of initial boundary nodes and interior nodes, create a tetrahedral mesh using Delaunay method.

6. Apply necessary post processing and clean-up operations.

Figure 2.1 shows a flowchart of these steps.

2.3 Algorithm

Essentially, we "pixelize" the domain of interest, i.e. saturate the bounding box of the region to be meshed by a grid of pixels or voxels. This strategy provides a fast and

clean access to the domain and to the necessary operations in order to appropriately deploy spatially dependent nodes. The spacing between deployed nodes is governed by a user-defined density function. The grid of voxels is a template that envelopes the domain of interest in a matrix format. Each pixel can be identified by 3 integer numbers representing the indices of the aforementioned matrix in 3D. Depending on the mesh sizing function, pixels in proximity of initial boundary nodes are tagged as unavailable. That is, a buffer zone about each physical boundary node is established. The extent of the zone is spatially dependent. All unprocessed pixels within that zone are tagged as unavailable. The algorithm advances inward from the boundary. When an unprocessed pixel is encountered, a node location is tagged. The mesh density at that pixel location is determined and a buffer zone about this node-tagged pixel is established. The process continues until there are no unprocessed pixels. This inward direction motion is similar in spirit to an advancing front[66], paving[14, 8], or offset normal strategy[54]. However, no processing of working surfaces or partial meshes are required. This node deployment strategy is robust and fast. Once all pixels are processed nodes are created at the node-tagged pixel locations. The 3D matrix grid is removed from memory. The resultant nodes have adjacent neighbors spaced according to the user-specified mesh density. The nodes are passed to a 3D Delaunay routine with tetrahedrons returned. A simple centroid check of the tetrahedrons eliminates most elements external to the physical boundary. Finally, the mesh is passed to our last resort algorithm[44] to recover all original physical boundary surfaces.

2.3.1 Description

As mentioned previously, performing basic geometrical predicates on floating-point variables introduces round-off errors as well as heavy loads on the CPU. One of the main objectives of using a template matrix is to avoid such complications. Mapping all the nodes, including boundary nodes, of the domain to three integer values helps us achieve the objectives. Figure 2.2 shows a sample 2D template matrix overlapped on a given boundary. The spacing dx, between voxels or pixels of the template is calculated based on the minimum desired edge size and the resolution of the template. The resolution basically defines how many voxels should represent the minimum desired length. For the examples presented in this work our resolution was set at 2.

$$dx = \frac{minimum\ edge\ size}{resolution} \tag{2.1}$$

The size of the template is obtained based on the bounding box of the domain to be meshed.

$$\begin{aligned} N_{row} &= \texttt{ceil}(X_{max} - X_{min}) + 2 \\ N_{col} &= \texttt{ceil}(Y_{max} - Y_{min}) + 2 \\ N_{pln} &= \texttt{ceil}(Z_{max} - Z_{min}) + 2 \end{aligned} \tag{2.2}$$

Adding +2 to the size of matrix ensures that the entire domain is engulfed by the template P which is a 3D matrix of size $P(N_{row}, N_{col}, N_{pln})$. Before proceeding fur-

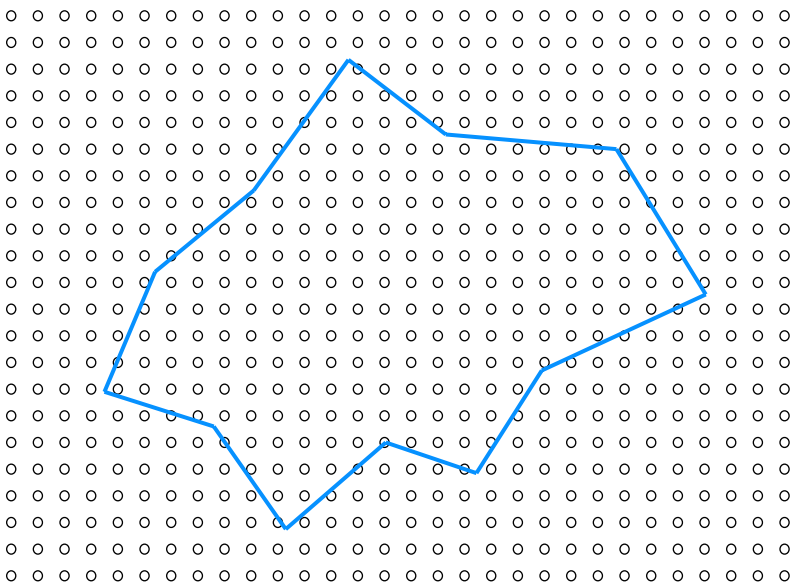


Figure 2.2: A template grid overlaid on a 2D domain

ther, we temporarily perturb the coordinates of boundary nodes in all three directions by a very small amount to ensure that the coordinates are not integer multiples of the pixel spacing. This infinitesimal node offset eliminates point-in-polyhedron inclusion tests issues. We use an extension of Jordan Curve Theorem[55] to 3D in order to determine interior/exterior status of placed points for the element formation process. Using Jordan's method, we cast a ray along a pixel line and count the number of crossings it has with the boundary of domain. Therefore, this method is heavily dependent on ray-triangle intersection algorithms such as Moller's [74]. To implement this algorithm in a robust manner, singularity and ray-triangle special intersection cases need to be avoided. If a ray intersects a given triangle within the area bounded by it (but not on the perimeter), the intersection is a valid one to be considered in Jordan Curve theorem. Otherwise it can not be determined if a valid crossing of ray

through boundary has occurred. However, we can use the following lemma to account for intersections along a boundary edge, but not at an edge vertex.

Lemma 1. *Assume ψ is a closed oriented triangulated surface and ρ is a cast ray intersecting at least one of the edges of ψ. The ray ρ is said to be crossing the domain bounded by ψ if both normal vectors of triangles sharing the crossed edge oppose OR follow the direction of ρ, i.e:*

$$(\overrightarrow{\rho}.\overrightarrow{n_1})\,(\overrightarrow{\rho}.\overrightarrow{n_2}) > 0 \tag{2.3}$$

where $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$ are the normal vectors of the two triangles.

Recall that we perturbed boundary nodes such that they were not integer multiples of the pixel spacing. Therefore, the ray will not intersect boundary vertices. It is necessary to emphasize that the perturbation process is independent of how our algorithm handles the template and it will not affect interior node deployment process. For every boundary node $p(x, y, z)$ we make sure that the following inequality does not hold true in all three directions:

$$|\texttt{mod}(x - X_{min}, dx)| < \epsilon \tag{2.4}$$

If it does hold true then the value of x is slightly perturbed using:

$$x := x + \epsilon \cdot \texttt{rand}() \tag{2.5}$$

In equations (2.4) and (2.5) $\texttt{mod}()$ is the remainder function, ϵ is a tiny value and

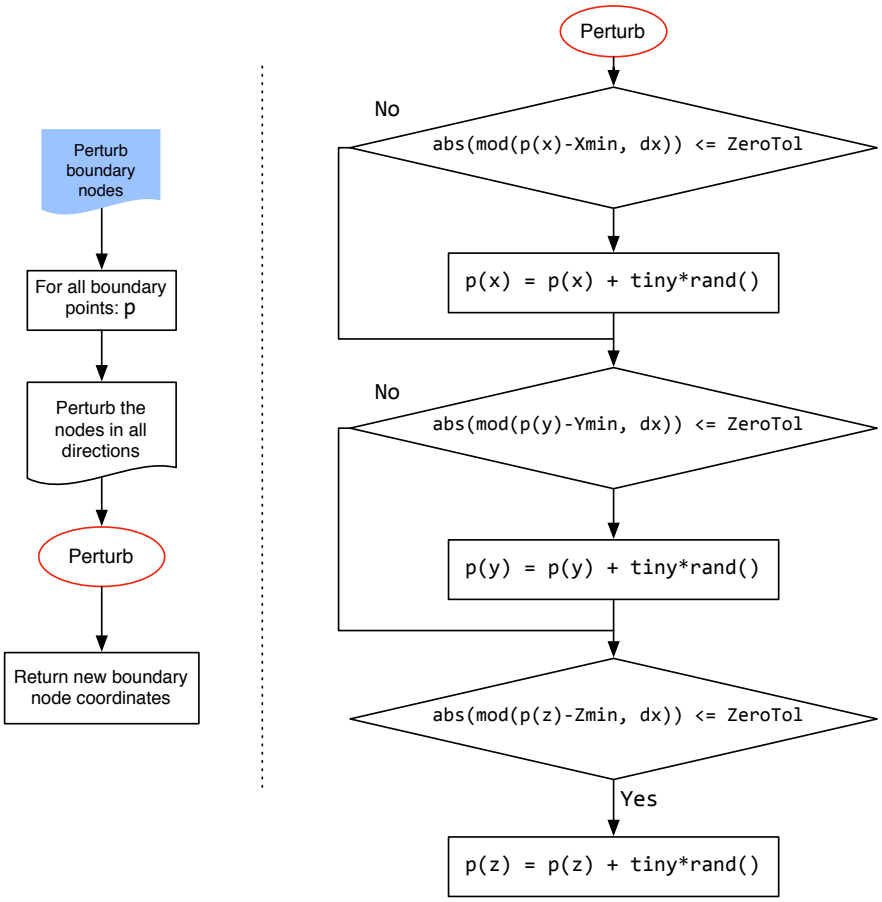


Figure 2.3: Flowchart of perturbing the initial boundary nodes in 3 directions

`rand`() is a random number generator function between 0 and 1. Figure 2.3 shows the flowchart for the above step.

Given that the domain of interest is enclosed by a triangulated surface mesh, we can identify the pixels that are closest to each boundary node and tag them as such. This can be done without using any "nearest-point-search" algorithm which would require floating-point calculations:

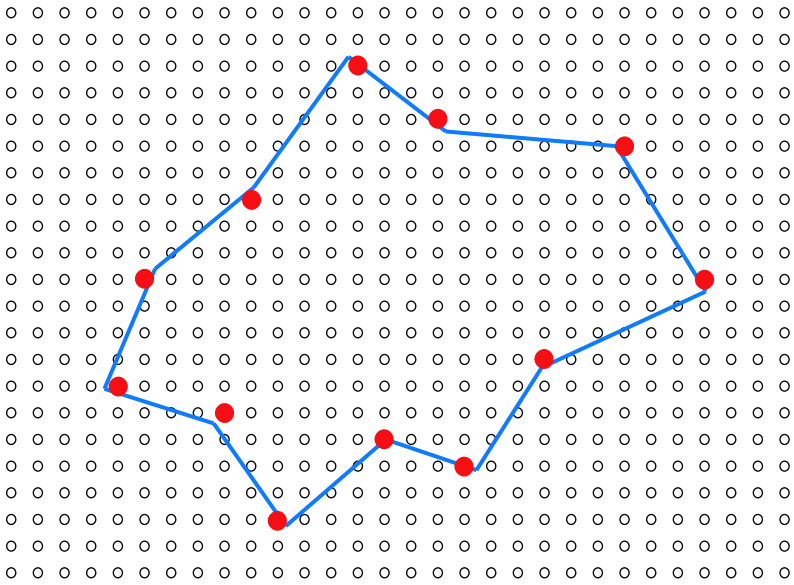


Figure 2.4: Finding closest pixel to each boundary node.

$$
\begin{aligned}
XI &= \texttt{round}(x - X_{min})/dx + 2 \\
YI &= \texttt{round}(y - Y_{min})/dx + 2 \\
ZI &= \texttt{round}(z - Z_{min})/dx + 2
\end{aligned}
\tag{2.6}
$$

Set of equations in 2.6 provide a fast mapping function to access $p(x, y, z)$ by reading the content of $P(XI, YI, ZI)$, Figure 2.4.

Based on the density requirements of the final mesh, there is an optimum distance from each boundary node that no other node can occupy. The distance can be acquired either by the user-provided density function or calculated based on the size of the triangles in the neighborhood of the boundary node in question. The

algorithm translates this distance in terms of number of pixels and accordingly all the surrounding unprocessed pixels of $P(XI, YI, ZI)$ within the given reach would be tagged as 'non-usable' or buffer. We refer to that set of voxels as the *buffer zone* of $P(XI, YI, ZI)$.

This process is repeated for all boundary nodes to establish a base layer of nodes that the rest of the mesh will be built upon. One can envision scenarios in which buffer zones of the 3 vertices of a triangle do not fully cover the span of the surface triangle. This happens when the size of initial triangle is much larger than that of the user-defined density requirements for the three vertices that define the triangle. In those cases, the buffer zone of the 3 vertices does not completely enclose the area of the triangle and "buffer holes" could exist. As stated in 2.3, algorithm places a node as soon as an unallocated pixel is available; thus these type of boundary triangles are susceptible to having nodes very close to their surfaces. To avoid this situation the following measures are taken for all the boundary facets.

Illustrated in 2D for simplicity reasons, Figure 2.5(a) shows a boundary simplex (an edge) with its nodes $N1$ and $N2$. The length of the boundary edge is greater than $R1 + R2$. Consequently, the buffer zones from two nodes do not cover the entire length of the edge which leaves a hole in the buffer. To resolve this issue, each end of the boundary edge is extended according to local density size and an area in 2D (a volume in 3D) is extruded. All the unallocated pixels within the enclosed region are tagged as buffer ensuring this edge (or surface) is sealed, Figure 2.5(b). Fig 2.6 shows a 2D boundary example whose boundary nodes are mapped to matrix world P and their buffer zone is established but contains some *buffer holes*.

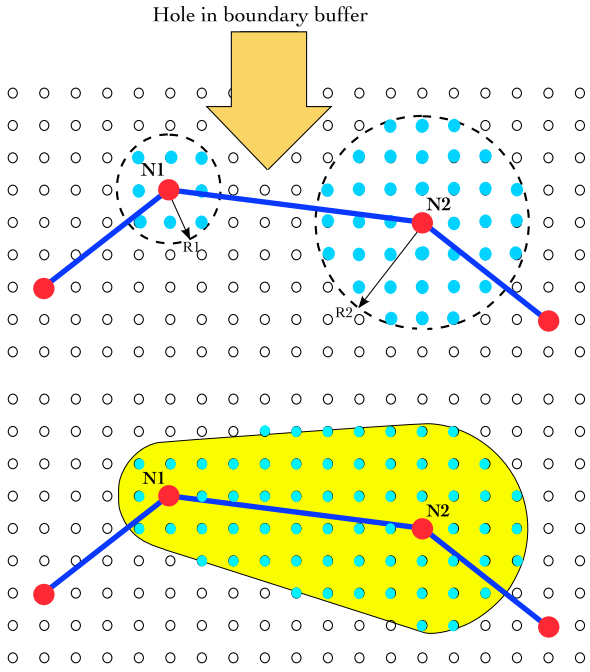


Figure 2.5: (*top*) 2D Example of a "hole" in the boundary buffer zone. (*bottom*) Vertex-dependent extended buffer zone of the boundary simplex $\overline{N_1 N_2}$

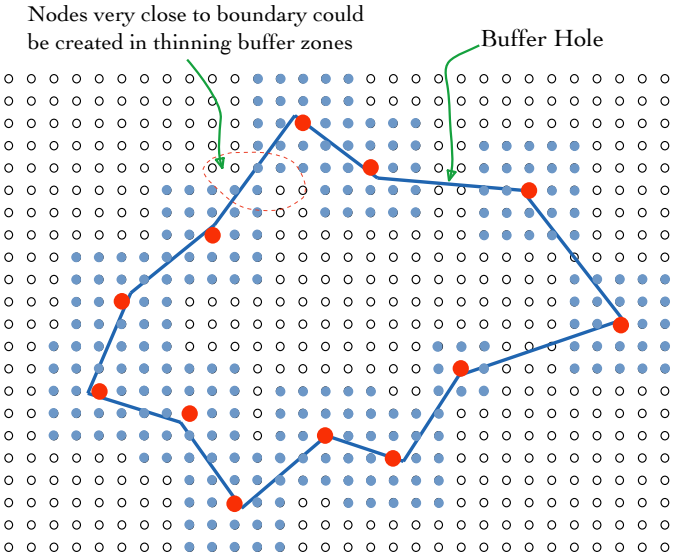
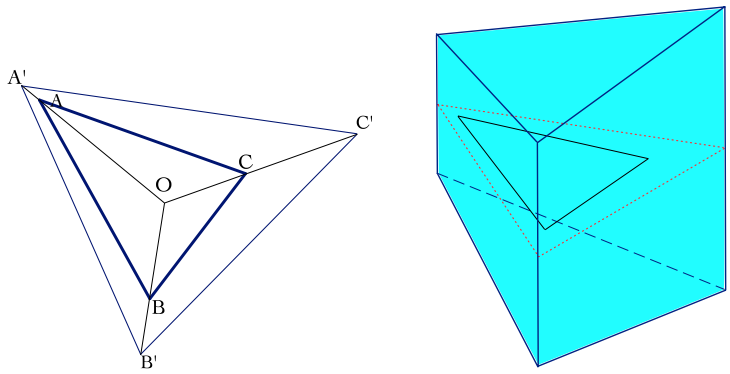


Figure 2.6: Issues related to boundary buffer zone if it were solely based on each single vertex.



(a) (b)

Figure 2.7: 3D Prism construction: *(a) Non-uniform scale of the boundary facet (b) Construction of prism using the scaled facet as the mid-plane.*

The same can be applied in 3D. Every boundary triangle associated with a buffer hole is non-uniformly scaled with respect to center of incircle with a scale factor of d_i where d_i is the desired edge size at triangle's vertex i. Figure 2.7(a) shows an expanded triangle in which vertex C' is obtained by:

$$\overrightarrow{OC'} = \left(1 + \frac{d_C}{\|OC\|}\right) \overrightarrow{OC} \tag{2.7}$$

After expansion, an imaginary prism-like enclosure is constructed around the facet using the triangle as the mid-plane of the prism, Figure 2.7(b). The axis of the prism is along the normal of the facet. The extrusion height of the prism on either side of the facet is again based on d_i, Figure 2.7(b). All unallocated pixels trapped within the prism are tagged as buffer zone which seals any potential buffer holes. This step ensures that no interior nodes are created in immediate vicinity of the boundary facets. Figure 2.8 shows the state of the template after this step.

Having the boundary buffer zone completely sealed prevents node creation closer

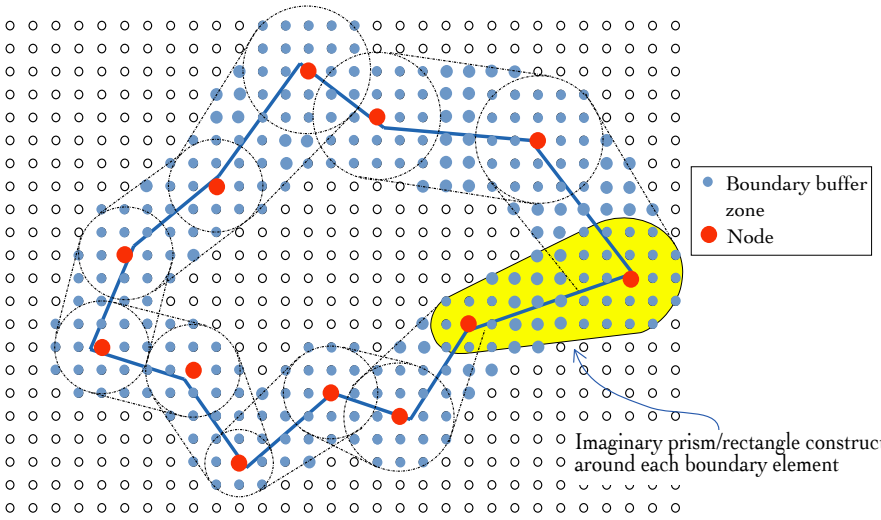


Figure 2.8: Sealing the boundary buffer using constructed prisms.

than required edge sizes in the final mesh. At this stage, all the pixels of the template are traversed and their status is checked. If a pixel is either a buffer pixel or a node-tagged pixel of any sort (boundary node or interior node) it is skipped. Otherwise, as soon as an available pixel, $\rho_{i,j,k}$, is encountered , it is tagged as a node location and its neighbor voxels $\mathbb{N}$ are defined as buffer zone.

$$\mathbb{N} = \{v \,|v \in P,\ \|v - \rho_{i,j,k}\| \leq d_{local}\} \tag{2.8}$$

where d_{local} is the desired edges size at the current location (i, j, k). Note that $\| \|$ operator measures the distance between two points by simply counting how many pixels separate them in all 3 directions; therefore no squaring or square root calculations are performed. If any member,v_i, of $\mathbb{N}$ is identified as a node, as oppose to buffer zone or unprocessed pixel, its status will not be altered. This can happen when the local density requirement of $\rho_{i,j,k}$ is coarser than that of v_i. By not toggling the

status of v_i the algorithm is essentially favoring a denser element formation over a coarser one, Figure 2.9.

After all the pixels of P are examined and tagged either as *"possible node"* or *"buffer zone"*, the algorithm calculates the actual coordinates of the pixels that were tagged as potential interior nodes:

$$Node_{int.}(x, y, z) = \begin{pmatrix} (i-2) \cdot dx + X_{min} \\ (j-2) \cdot dx + Y_{min} \\ (k-2) \cdot dx + Z_{min} \end{pmatrix} \tag{2.9}$$

The above set of nodes can potentially contain locations that are outside the original physical boundary and will be removed, Figure 2.9(b). The Jordan method is used to perform point in polyhedron inclusion test for all nodes created as described in section 2.3.1. Note that since we have perturbed the original boundary nodes, it is not possible for the ray to intersect a triangle exactly at one of its vertices and therefore Lemma 1 can be used. After this step, all the interior nodes along with original boundary nodes are passed to the tetrahedral element formation using Delaunay method, Figure 2.10.

As discussed in [44] there are different implementations of the Delaunay algorithm but all share some common characteristics:

- The final mesh is the convex hull of input domain boundary.
- Original boundary facets are not preserved in the tetrahedral mesh
- There can exist some poor quality tetrahedrons in the mesh.

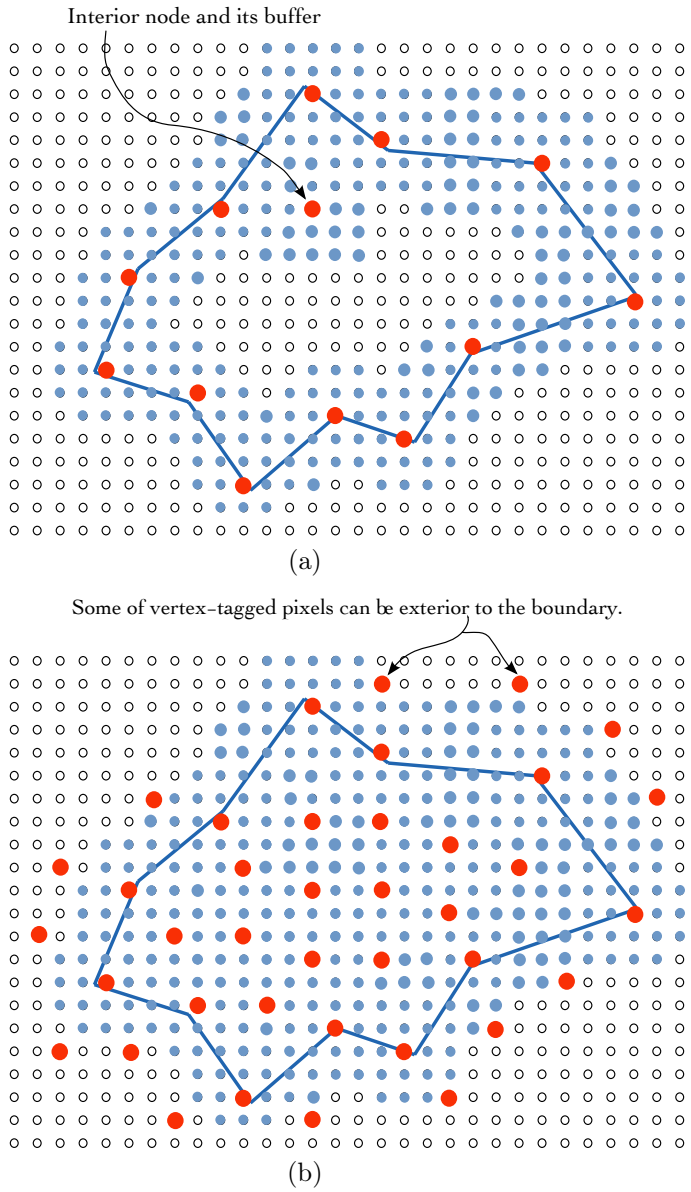


Figure 2.9: Placing the nodes in blank template locations: *(a) Finding the first empty pixel and creating a node. (b) Covering the entire domain using empty pixels.*

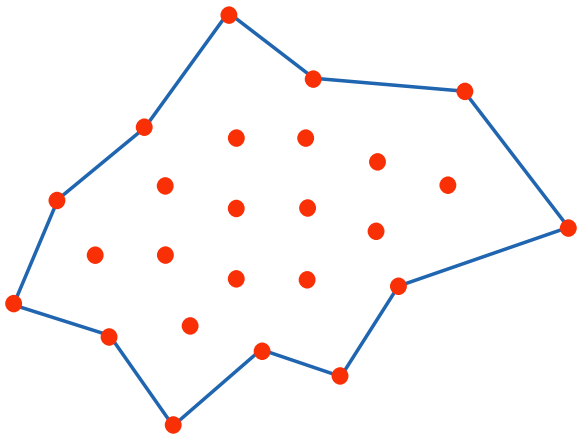


Figure 2.10: Final result of placing nodes in a 2D example

We addressed the last two items in the previous paper by performing local topology transformations without introducing new nodes to the mesh. The algorithm carves out small dents in missing facets regions and fills them with a new sub-mesh recovering the face. For cleaning the convex hull in the final mesh, the algorithms flags all the tetrahedral elements that at least two of their vertices is a boundary vertex. If the centroid of the examined element fails the inclusion test that tetrahedron will be erased from the database.

2.4 Results & Discussion

We used the algorithm to create meshes for human brain and breast models. Both input boundary surfaces were constructed from segmented two dimensional MR scans of the organs using multiple material marching cube M3C[116]. As outlined in Figure 2.1, boundary recovery and cleaning processes were performed on all examples. For

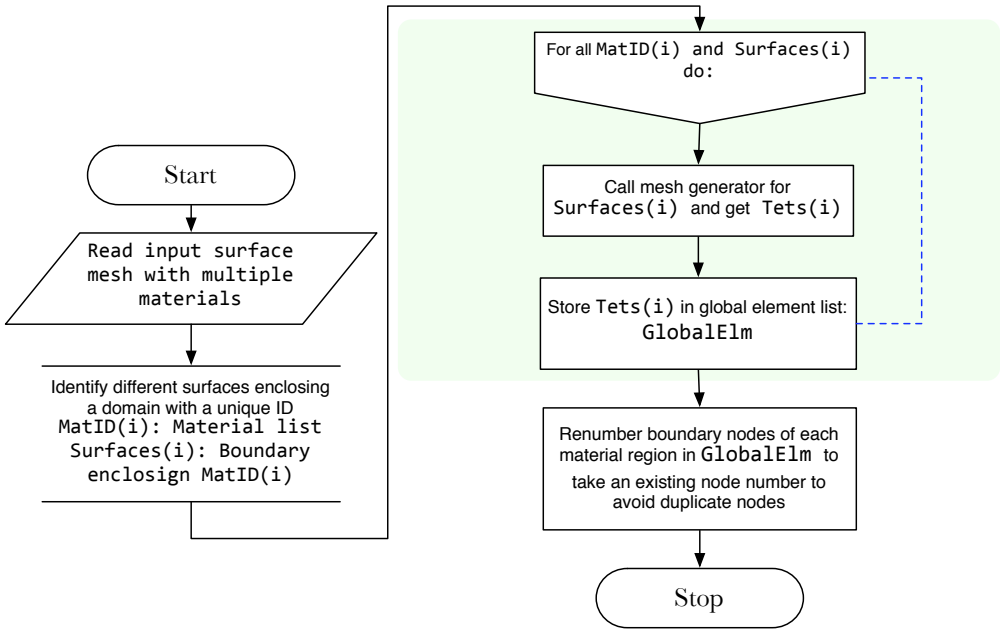


Figure 2.11: A simple algorithm to utilize this mesh generator in multiple material region scenarios.

complete clarity of the mesh generation details, all the illustrated examples are for single material volumes. However, our algorithm can be easily used to create meshes for various regions with different material ID and then concatenated to form the final mesh with minimal effort since the physical boundary faces separating different regions are restored with fidelity, as shown in Figure 2.11. The algorithm was implemented in MATLAB (a strong development environment albeit with compromised execution performance) and was run on a machine with Mac OSX 10.5 as the operating system with a 2.8GhZ Intel Core Duo CPU and 2GB of DDR2 RAM. The relationship between time required to generate a mesh and number of final tetrahedron is of $O(n)$, Figure 2.16.

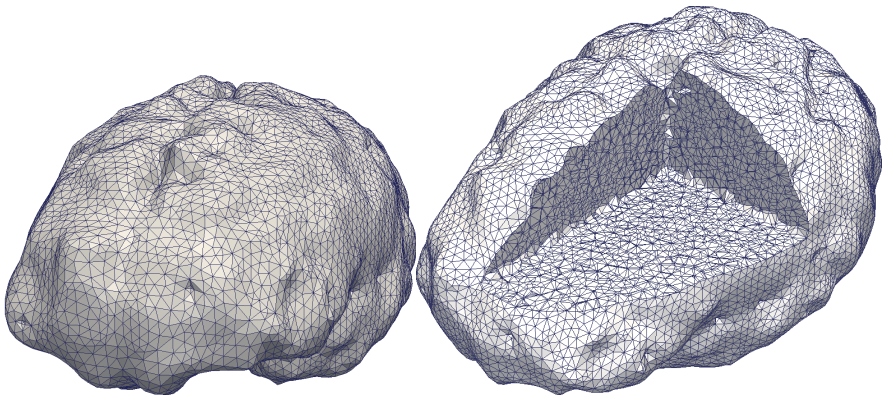


Figure 2.12: Surface depiction of white and great matter of human brain. It was constructed using a marching cube algorithm on Harvard brain repository. A cross section of the generated solid mesh is also shown.

2.4.1 Human Brain

Figure 2.12 shows multiple slices of a human brain MR image volume which is publicly available from Harvard Brain Segmentation Repository[16]. A segmented volume set for white and grey matter was also provided at the Harvard Brain Segmentation Repository. A boundary mesh was reconstructed from these 2D medical image slices using M3C. The surface mesh contained 22,964 surface triangles and 11,476 nodes. The final volume mesh retained all physical surfaces and contained 250,416 tetrahedral elements. The cutout view of Figure 2.12 demonstrates the interior tetrahedral elements and their intersection with profiling planes.

Another human brain [51] input surface was constructed using the M3C code is shown in Figure 2.13. This mesh was generated using a 17,312-face surface mesh which resulted in a solid mesh with 199,129 tetrahedron and 35,686 vertices.

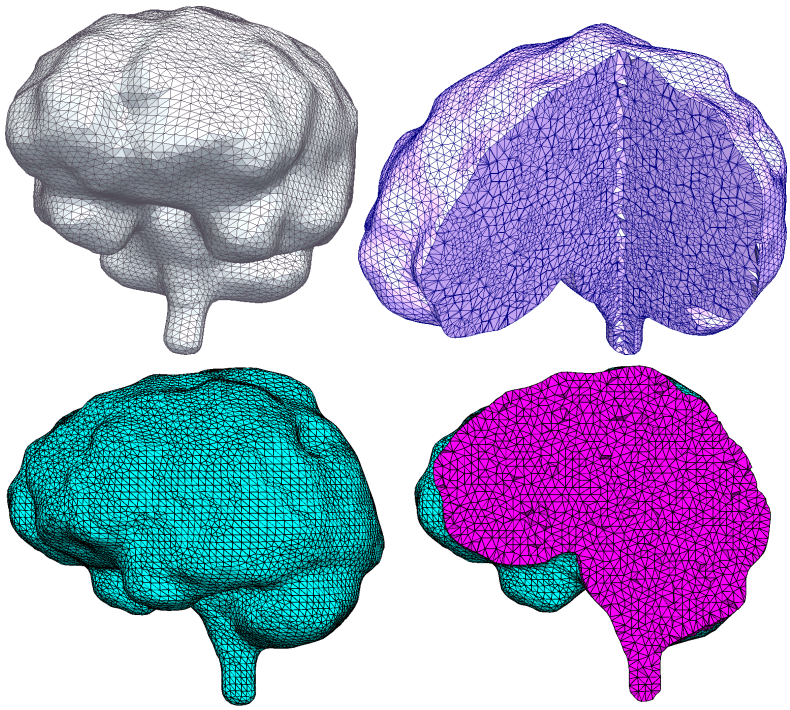


Figure 2.13: A human brain meshed using proposed template-based algorithm.

2.4.2 Human Breast

Using data from Dartmouth College's Alternative Breast Imaging project, a human breast surface mesh was constructed. The needle-type spikes or serrated dents seen in Figure 2.14 is result of imaging probes pressing on breast tissue and deforming it, Figure 2.15. The probes were used to receive and transmit near infrared (NIR) waves [117]. The triangulated surface mesh for this model has 9,864 faces and 3,014 nodes. The final solid mesh contained 29,719 tetrahedral elements.

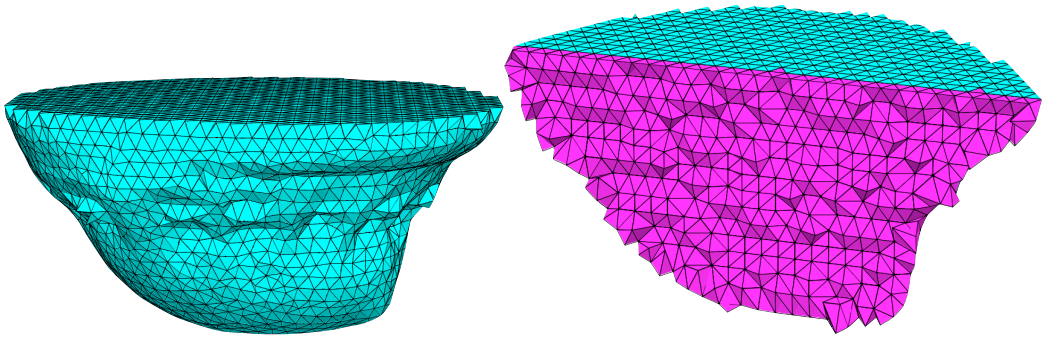


Figure 2.14: Solid mesh of a human breast constructed from 2D images obtained from Alternative Breast Imaging Techniques.

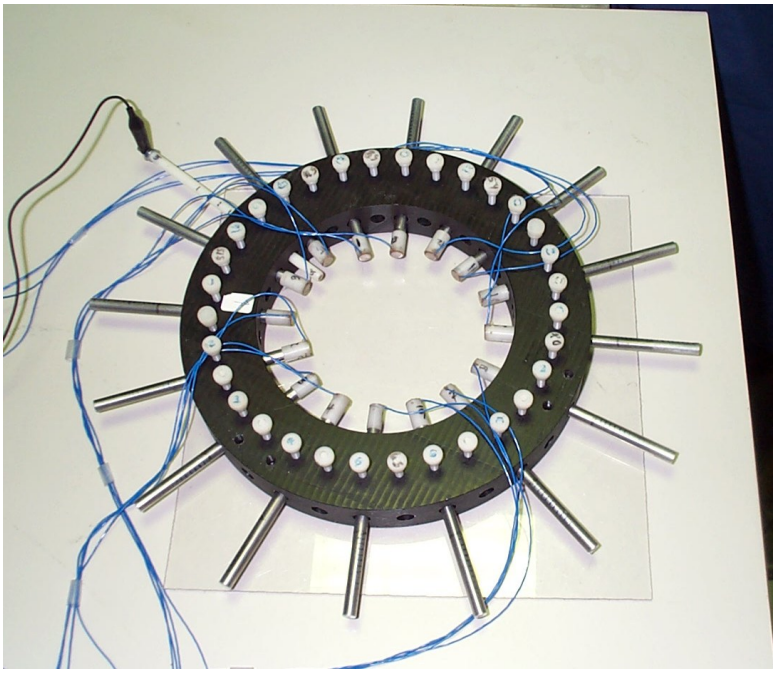


Figure 2.15: Apparatus used in *Near Infrared* imaging of the breast. The spikes on the exterior of the breast mesh in Figure 2.14 are caused by the transreceivers on the ring.

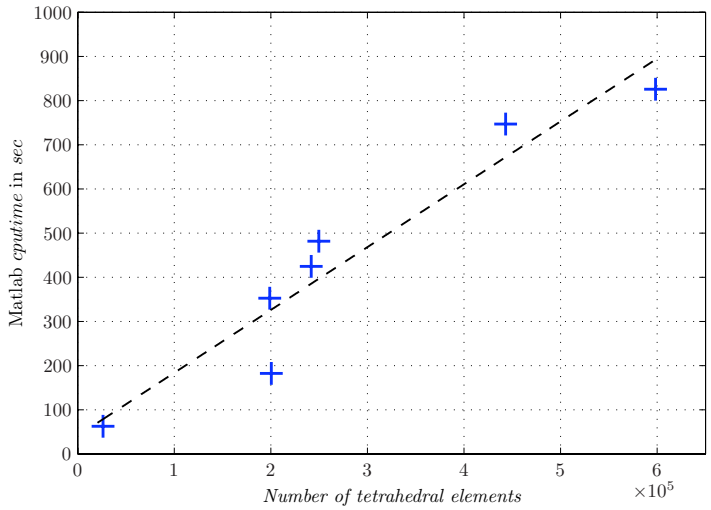


Figure 2.16: CPU-time and number of generated elements are linearly proportional.

2.4.3 Timing and comparison

Table 2.1 shows the performance of the proposed algorithm prototyped in Matlab. Given the slower performance of Matlab versus a high level language such as Fortran or C/C++, the timings shown in the table are extremely superior to the timings of SPMESH method, a utility which is being used by researchers in Dartmouth College to produce high quality, multiple material tetrahedral meshes. SPMESH partitions the input domain using skewed bricks and then extracts tetrahedral elements based on desried sizing requirements. It is also capable of performing multi level refinements. This creates almost perfect tetrahedrons inside the domain. To achieve the same quality adjacent to boundary, SPMESH tries to morph the elements to conform to the boundary, which results in a mesh that is not fully constrained and at times sharp corners are rounded.

An accurate comparison would have been possible if both algorithms were im-

	Time (min)	# of tetrahedrons	Template memory in *kByte*
Brain 1	4:30	250,416	1,107.30
Brain 2	3:00	199,129	953.07
Breast	0:35	29,719	140.15

Table 2.1: Performance of mesh generator algorithm versus number of tetrahedrons.

plemented in the same platform. In order to demonstrate the performance of our algorithm a rough comparison was made by running the SPMESH on breast model. It took SPMESH about 12:00 minutes to create the same size mesh while our algorithm generated the mesh in 00:35 seconds.

On the other hand the memory footprint of our algorithm is minimal. The template matrix overlaid on domain can be stored in memory by saving multiple pixels information in a single byte, although our implementations uses one byte of memory for each pixel. Table 2.1 shows the memory used by our implementation to generate meshes in previous sections. It is obvious that template matrix memory can be released as soon as all the pixels are processed and tagged, and it is not needed for the topology formation.

The template based algorithm proposed in this chapter lays the foundation for creating tetrahedral meshes for complex biological models with great user control over sizing of the mesh. The mesh created by Delaunay method using the deployed nodes by this algorithm is ready for consequent post processing steps, which can include optimizing the mesh quality or recovery of original input geometry, if required. Majority of elements created using the proposed method have high quality and the

only type of compromised element present in the mesh is *sliver* tetrahedron (see Figure 1.3) which is result of Delaunay method. The mentioned post processing steps are not unique to our algorithm. Therefore in next chapter we introduce a new approach to recovery original input geometry and remove the sliver elements.

Chapter 3

Boundary Recovery using LAST RESORT

3.1 Introduction

Finite element method is a versatile tool that tackles various problems in multiple engineering fields such as structure design, Bioengineering, MEMS and Nanoscience [13, 45, 46, 85, 118]. Discretizing the domain of the problem is an essential step to FE method. Adaptive problem solving strategies frequently require automated (re)generation and modification of the mesh in order to optimize various parameters in the problem and solution. For productivity reasons a fully automated mesh generator is a must [91], and historically, problems encountered in the mesh regeneration process are far more difficult than the rest of simulation process [94].

Post-processing a mesh is often a necessary step to ensure element quality. This

stage is paramount to success of applications using the mesh. Each mesh can be examined for various quality measures such as element shape measures (minimum solid angle, radius ratio ρ and gamma coefficient γ [63]), regularity measures [21] and insphere deformation [60]. The accuracy and outcome of FE solutions rely on the quality of the mesh. For example dihedral angles of a tetrahedron in a mesh can have significant impacts on either precision of computations or the conditions of the stiffness matrix [31, 99]. Frequently, a finite element implementation requires the mesh to preserve the original defining boundary topology. This requirement is critical for adaptive remeshing of a sub-volume, or a multiple material volume coherently connected [97].

Delaunay triangulation is a favored approach in creating 2D meshes due to its inherent properties. For example, Delaunay triangulation tends to create well-shaped elements by maximizing the minimum interior angle over the domain [105]. The elements are also suitable for use in interpolation [99, 106] as well as refinement methods that can help generate meshes with provably good properties [25, 72, 27, 95].

The highly successful 2D Delaunay mesh generation strategies provided strong momentum for a 3D extension. Delaunay tetrahedralization has been used in [90] to simulate respiratory gating which helps diagnosticians overcome problems with artifacts in SPECT imaging method due to breathing motion. Sullivan and Zhang simulated 3D contaminant and fate transport of benzene for groundwater flow situations experiencing semi-discontinuous permafrost [107]. Quad-tree [119], Delaunay triangulation, advancing front [65] and sphere packing [101] are most common methods of mesh creation, out of which Delaunay is being used widely for establishing the

connectivity [64, 114, 10, 93, 112, 4, 67, 9]. Unfortunately Delaunay tetrahedralization lacks several of the inherent qualities noted in the 2D triangulation. 2D Delaunay maximizes the minimum angle in the triangulation [58], minimizes the largest circumcircle and minimizes the largest min-containment circle[1][22]. Unfortunately the first two properties can not be extended to higher dimensions, albeit the last one is proved to be a general property of Delaunay triangulations [82]. Additionally, 3D Delaunay triangulation does not always exist for every polyhedron [89] whereas all 2D polygons have a Delaunay triangulation.

Delaunay tetrahedralization can have undesirable side effects: poor quality elements and missing input boundaries. Different approaches have been developed to address these issues including introduction of new vertices at the centroids or the center of circumcircle of problematic elements or using advancing front methods [39, 66, 113]. Another significant difference is that in 2D the edge elements are preserved in the triangular mesh provided a constrained Delaunay was used. However, in 3D the triangular surfaces are not always guaranteed even if a constrained Delaunay is used [100]. This situation can raise problems for FEM applications. For instance, maintaining simplexes of the boundary is necessary for applying boundary conditions imposed by the application. Similarly, respecting the interface between two biological tissues with different diffusion coefficients, when solving a material propagation problem, is required.

Post-processing or *clean-up* of a mesh generally falls into two categories: smoothing and topological optimizations [78]. The former is relocating one or more nodes

[1]Min-containment circle of a triangle is the smallest circle that contains it and is not necessarily its circumcircle

in order to increase quality measures, while the latter is changing the connectivity of the nodes in order to form better elements. During last two decades, extensive work has advanced these post-processing tools significantly. [6, 34, 83, 41]

However, recovery of the original physical boundaries (internal and external) has remained illusive. This paper presents a successful strategy to recover all physical boundaries regardless of the original technique used to create the mesh. A natural underlying mechanism within this approach is the optimization of element qualities in the vicinity. This latter feature can be applied throughout the mesh.

3.2 Existing Approaches

As mentioned previously, boundary recovery is necessary for meshes created by Delaunay tetrahedralization and it can be difficult for certain domain boundaries [97]. Studies and research on boundary recovery mainly use two strategies in order to mitigate the problem: *(a)* Breaking up input boundary constraints and *(b)* a series of edge and face swaps/flips. In general there are three different approaches: conforming Delaunay, almost Delaunay and constrained Delaunay tetrahedralization [97].

- Conforming Delaunay

Conforming Delaunay has been presented in several works as a technique to recover the physical boundary topology [76, 20, 43]. In this method additional vertices are introduced to the mesh. The methods differ based on the interior node insertions locations and criterion to minimize the number of point insertions. These strategies do maintain the Delaunay property of empty circumspheres. Upon successful completion

of this procedure, the boundary facets are basically subdivided into several smaller ones. That is, the original surface facets have been subdivided. Consequently, they are not suitable for remeshing a sub-region of a domain. Additionally, these algorithms can create extremely small tetrahedrons. Liu shows some cases that fail the above method [59].

- Almost Delaunay

Several researchers retrieve the domain boundary by inserting additional points wherever a face of the mesh intersects a missing segment or an edge of the triangulation intersects a missing facet [47, 42, 114]. Again the faces are recovered as a union of smaller faces. If the input physical boundary was defined as a polygon and not necessarily as a single triangle, this technique is valuable. However, it still produces an inconsistent mesh if the recovery of the boundary was an interior boundary. Within this category two investigations have developed strategies to recover the boundary without introducing new nodes. Originally, Wu developed the first 'Last Resort' code in 2001 [115]. The system could recover the original boundaries, but the time complexities were considerable. Recently, a small polyhedron reconnection (SPR) operation was presented that reconstructs the original input boundary without adding new nodes [61]. This sphere-packing method packs spheres based on a mesh sizing function. Selection of this sizing function is important for their algorithm to work well. In this category 'Almost Delaunay', the Delaunay property is not fully preserved. The elements are split such that they introduce new vertices without removing existing faces and edges of the domain boundary. Although this method does not fully maintain Delaunay property, it is very common in practical applications.

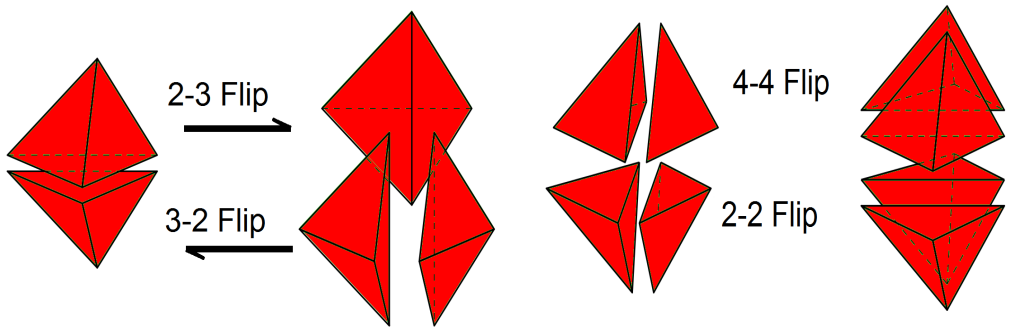


Figure 3.1: Typical connectivity transformations involving 2 to 4 elements.

- Constrained Delaunay tetrahedralization

In this method, after creating Delaunay tetrahedralization of the input vertices, which usually creates a convex hull encompassing the domain, the missing edges of the boundary are identified and new vertices are inserted to recover them. At this point the Delaunay tetrahedrons are replaced by *constrained Delaunay* tetrahedrons which recovers the missing faces. Usually these algorithms first generate a constrained triangulation of input boundary using different methods such as [19]. Then either a gift wrapping algorithm, sweep algorithm or incremental face insertion is used to construct constrained Delaunay tetrahedrons [96, 97, 108].

3.3 Algorithm

We present a different approach that does not involve introducing new vertices to the mesh. It is purely based on altering the topology in a limited section of the mesh. As stated previously, there are works that include changing the connectivity of the nodes in order to recover the boundary edges and/or faces [114, 26, 47]. Those methods

usually limit the alterations to edge swaps involving the missing physical face. For instance, Figure 3.1 shows common topological transformations involving edge flips. Some techniques insert new points to the mesh to force a different mesh configuration as a means of recovering the original physical face [76, 43], but frequently, this produces a composite surface and is not suitable for interior boundary recoveries. Our corrective approach is an edge/face swap strategy that expands the number of tetrahedrons traditionally involved in the connectivity alteration process. This feature results in multiple configuration options to recover the physical boundary facets simultaneous with an improved quality mesh in that particular region.

Our strategy carves out a cavity, caps the surface with the previously missing physical faces, if necessary, and subsequently fills the void with tetrahedrons. It is based on a fully exhaustive branching algorithm, hence dubbed LAST RESORT [115]. However, several algorithm improvements as well as performance optimizations have reduced the search time by orders of magnitude. Consequently, the LAST RESORT post-processing strategy is a practical plug-in for most mesh generation systems. Since all the possible topological configurations are examined, we not only recover missing physical boundaries but also have the potential to improve element qualities in the vicinity.

The next section delineates the LAST RESORT algorithm followed by its use to recover missing physical boundaries.

3.3.1 Last Resort

Modifying a sub-volume of a mesh is always in high demand as a post-processing stage. It can be used to recover original boundary topologies of a domain or help improve the quality of a mesh. A few degenerate elements, in an otherwise quality mesh, can compromise the reliability and accuracy of FEM results. Re-meshing just a sub-volume of that mesh can avoid repeating the entire meshing process and save computational resources. The LAST RESORT algorithm performs this task. Consider an interior element of compromised quality. The routine identifies the element, deletes it, and all neighboring tetrahedrons that share a common node. This carved out sub-volume creates a cavity that is closed. Each cavity shell triangle is considered an active surface initially. The system considers *all* possible tetrahedrons that can be formed using an active cavity shell facet and selects the best one based on a quality criterion. This tetrahedron will, at minimum, retire one of the original boundary's facets and potentially introduce three new active shell faces. This iterative process continues until the sub-volume is filled with consistent, conforming tetrahedrons.

Simply searching for the optimal solution using this method is inefficient. If we assume every N face can produce $N-1$ elements at every level, then the number of possible solutions to be tested is in the order of $O(N!)$. In following section we will show how to drastically improve the efficiency of algorithm by intelligently choosing which route to take in order to obtain an optimal solution in less time and make the algorithm a practical approach for mesh improvement problems.

3.3.1.1 Description

We describe the algorithm for a 3D case but the example illustrations are in 2D for clarity purposes. Assume a part of a mesh contains a low quality element $q \leq \eta_e$, Figure 3.2(a). We create a closed cavity by removing the low quality element (marked as X) and its surrounding elements (marked as Y), Figure 3.2(b). In 3D counterpart, this will be a closed surface, Ψ_i. The algorithm continues with the boundary surface enclosing the sub-domain to be meshed. This boundary can also include scattered interior points which will be used in the meshing process. All the entities of the closed cavity are considered active 3D faces (2D edges) or nodes (Figure 3.2(c)). Assume a boundary surface of N_f faces and $N_{scattered}$ interior scattered points or $n = N_f + N_{scattered}$ total nodes. Given that there exists n points in the original cavity, we can form m_i different tetrahedrons using face $F_i (1 \leq i \leq N_f)$ and another point (Figure 3.2(d)), where

$$m_i \in \{1, ..., n-3\} \tag{3.1}$$

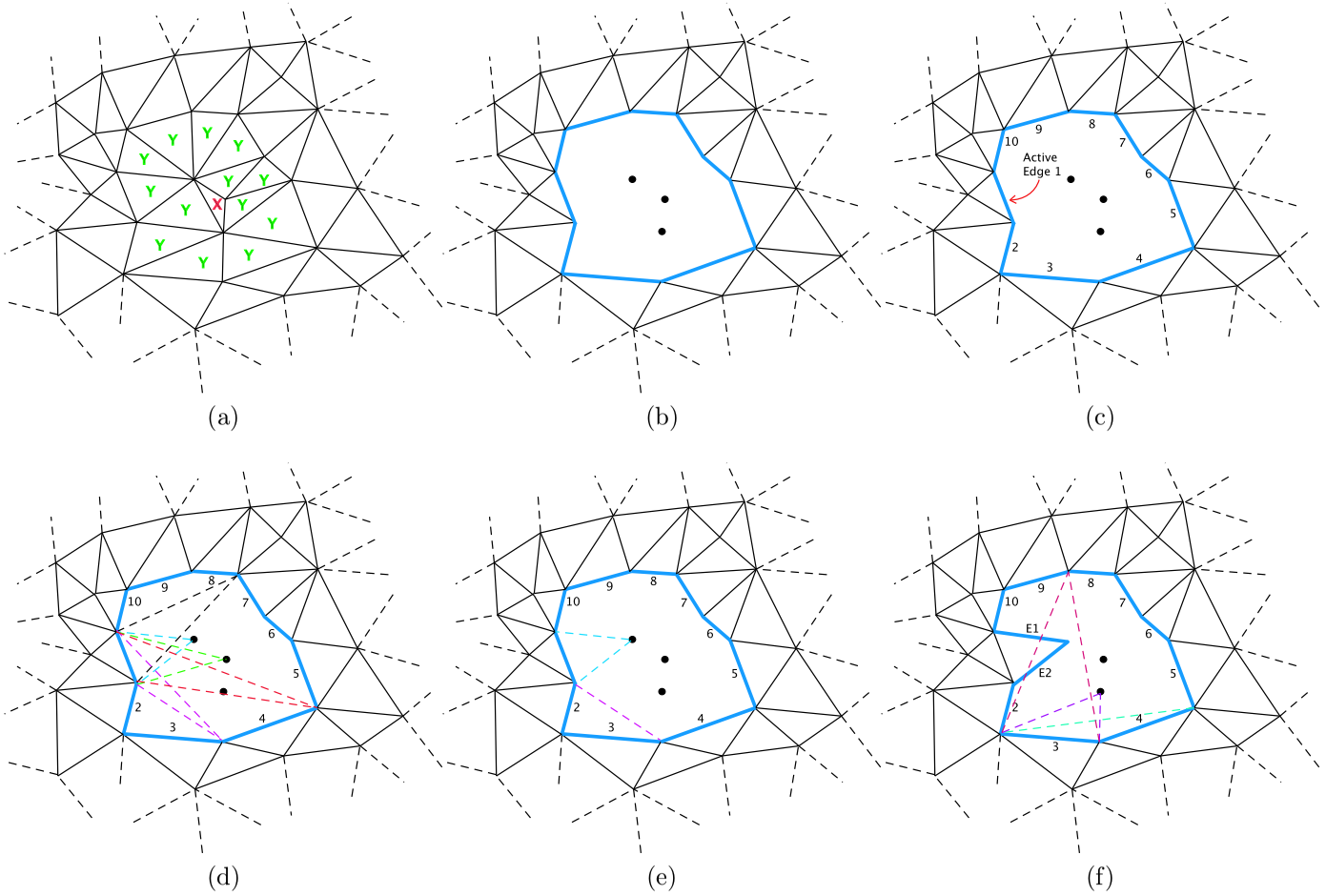


Figure 3.2: (a) Original mesh with a low quality element. (b) Carving out all the elements containing any vertices of the bad element. (c) Simplexes (active edges and isolated nodes) to be used for triangle formation. (d) Different possibilities of element formation, of which some are not viable due to either quality or geometry constraints. (e) Insertion of any possible element might introduce 1 or 2 new active edges. (f) Placing the new element creates a new set of active edges and the process continues by using another edge from list of active simplexes.

As shown in Figure 3.2(d), not all m_i possible formations are viable since some do not meet the quality criterion or their formation invalidates the continuity of the mesh. Next, using another active face F_{i+1} in the current boundary surface Ψ_i, we examine how many valid tetrahedral elements, whose quality is greater than η_e, can be formed: m_{i+1}. If $m_{i+1} < m_i$, the algorithm discards face F_i as a potential candidate for generating a tetrahedron at that level and substitutes it with F_{i+1}. Note that as in Figure 3.2(e), some of element formations might produce different number of new active edges or active faces in 3D case. The iteration continues and discards each face F_j if $m_j \geq m_{min}$, where

$$m_{min} = current\ minimum\ number\ of\ possible\ tetrahedrons \tag{3.2}$$

The result is a face with the least number of potential offspring tetrahedrons, out of which the best quality tetrahedron is chosen (Figure 3.3). The algorithm then inserts that tetrahedron into the domain causing removal of at least one boundary facet and creating potentially three new active faces. This process generates a different sub-volume encapsulated by a new boundary surface ψ_i, on which the same aforementioned process can be performed recursively (shown for 2D in Figure 3.2(f) and in Figure 3.4 for 3D case.)

As Figure 3.2 shows, removal of elements can leave behind some internal nodes within the carved out region that do not connect to any other simplex. This makes no difference in how LAST RESORT makes use of them. All the simplexes involved in the carved out region are examined in the process of forming all possible tetrahedrons regardless of their isolation state and no element is formed if it contained any node,

including the isolated ones.

This process continues until there only exists four faces which form the last tetrahedral element and the entire domain is populated. Once one complete solution is achieved, we can trace back our steps to mother face set Ψ_i and try a different route. Upon successful completion of each try, the algorithm compares the quality of all the solutions, either based on minimum element quality or an average quality of elements, in order to choose the desired mesh. This can also be done exhaustively to examine all possible connectivities, but this option is not realistic since it can be time consuming. Figure 3.2 shows an example of the steps described above, depicted in 2D for simplicity in illustration.

If in any given sub-surface ψ_i, one of the triangular facets, F_{failed} , fails to produce any acceptable tetrahedral element, that step is considered void and none of the other available facets of ψ_i are examined (Figure 3.5). At this point, the algorithm goes back one step and tries the previous sub-surface Ψ_i, using a different triangular facet to form an element (Figure 3.6).

If we had used other faces of a failed ψ_i and created a different tetrahedron path, the resulting sub-surface still contains F_{failed} and eventually it must be connected to one of the available points in the current point set but we have already established that such connectivity was not acceptable. Algorithm 3.1 flowcharts the LAST RESORT process. As discussed previously, performing an exhaustive search along only one branch of the tree in Figure 3.3 is of the order $O(N!)$. Therefore the upper bound for number of operations necessary for an exhaustive search can be expressed as:

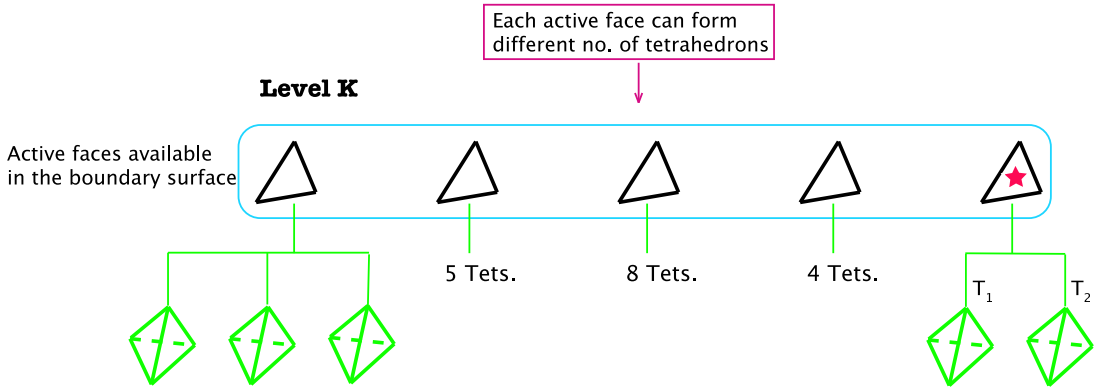


Figure 3.3: Each active triangle in Level K has the potential of producing different number of tetrahedrons. Every possible tetrahedron is examined for quality. The active face with the least number of possible tetrahedrons is chosen to form an element.

$$\sum_{i=0}^{N}(N-i)! \tag{3.3}$$

However the selective approach for the path descent down the search tree will reduce the upper bound of required operations to:

$$\sum_{i=0}^{N-1}(N-i).\,(N-(i+1)) \tag{3.4}$$

where N is the total number of original active faces in the carved out surface Ψ.

3.3.2 Boundary Recovery

As mentioned previously, our goal is to recover the original surface topology of any given tetrahedral mesh regardless of how it was created. This task requires having the original boundary surface Θ, and the mesh Γ. The idea is to identify a missing face and remove all the elements that are connected to its three vertices, creating an

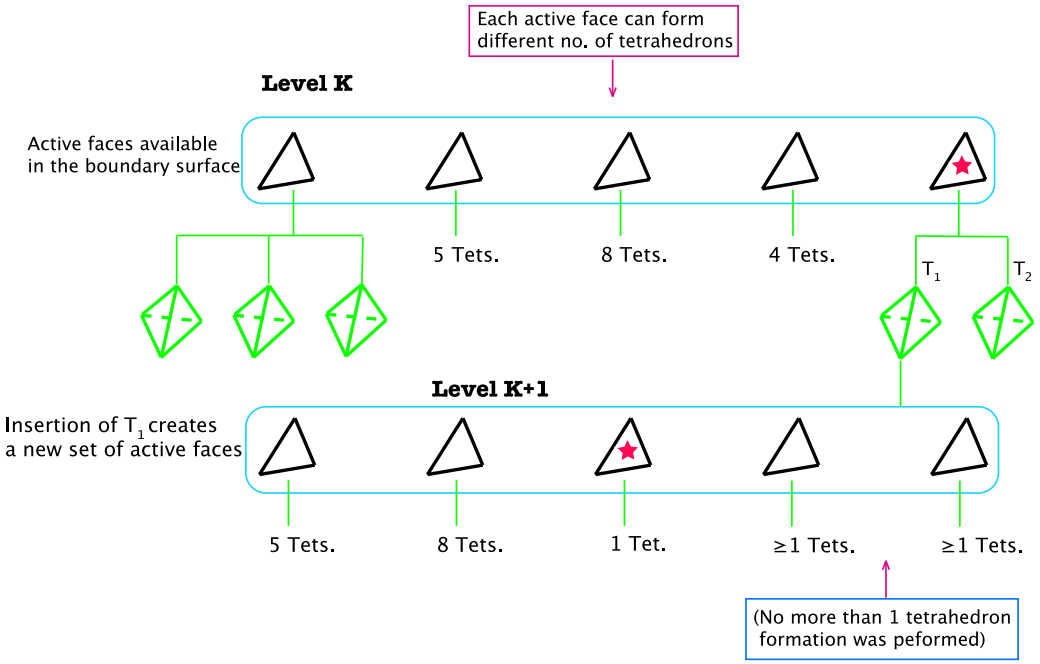


Figure 3.4: Insertion of tetrahedron T_1 to the sub-volume will create a new set of active faces. This moves the algorithm one level down $(K+1)$. The new active triangles are now ready to be examined for their possible tetrahedron formations.

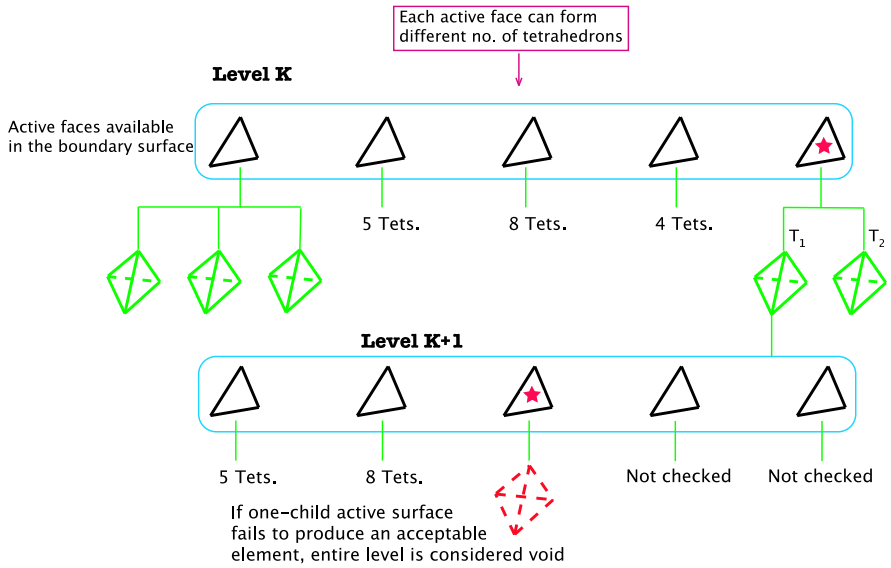


Figure 3.5: If one of the active faces fails to produce an acceptable tetrahedron, the algorithm will not check additional faces on that level and that branch is tagged as broken.

open cavity in the mesh. The cavity will not be sealed since the elements associated with the missing face nodes were removed. The missing face is recovered and added to the active cavity surface shell and additional triangles are added to seal the cavity if necessary. The resulting closed surface is now ready to be passed to our LAST RESORT operation to create new tetrahedral elements that can be directly inserted into the parent mesh.

As a preliminary step, we remove all elements of the parent mesh whose centroid fails the Θ inclusion test. This rudimentary constrained Delaunay recovery is a necessary step for meshes created by Delaunay method as it produces a mesh covering the convex hull of Θ. All boundary faces of the mesh Γ are identified. These faces are contained in only one tetrahedron of the same material type. We refer to this set

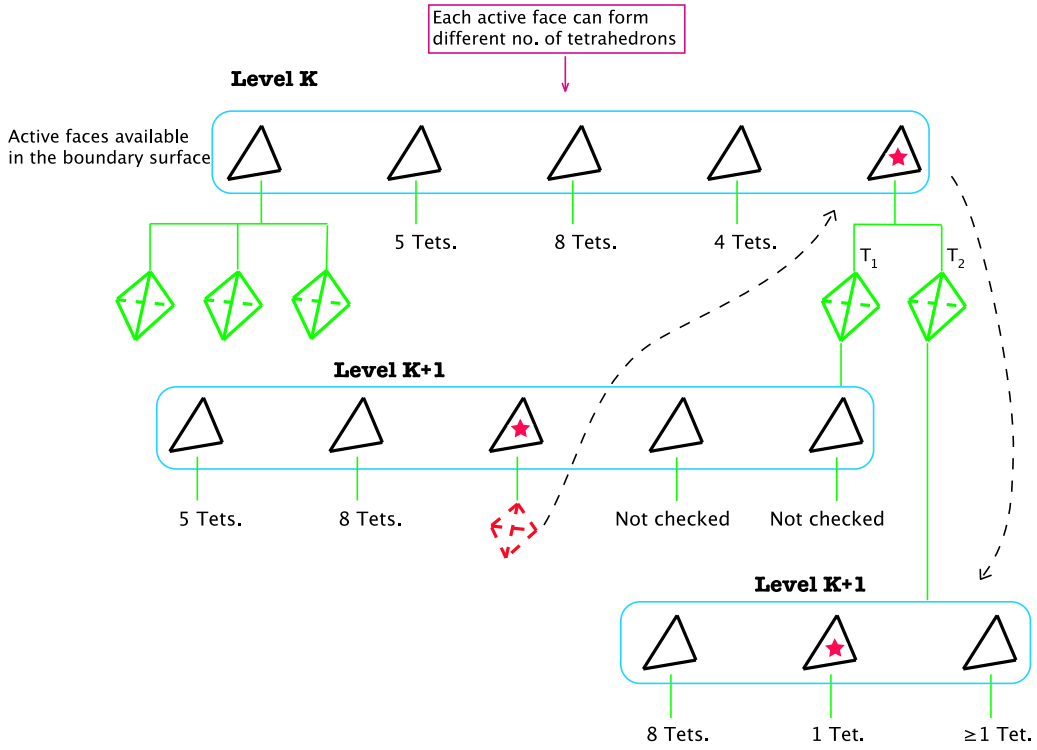


Figure 3.6: A *broken* branch forces algorithm to consider other active faces in previous level (K). Here tetrahedron T_2 from previous level is considered and a new level of active faces is created in the same way as Figure 3.3.

of faces as numerical boundary faces Ω, as opposed to the physical boundary Θ. The faces in Θ are compared against the Ω list to identify the missing physical faces of interior or exterior boundaries.

For every missing face M_i, we identify its 3 vertices and tag all the tetrahedrons whose vertex list contains at least one of the 3 vertices. Concatenating the tagged tetrahedrons creates a shell ψ_i which is not necessarily a closed surface, meaning that there are edges of triangular facets on ψ_i that only belong to one boundary face. Note that ψ_i is a piecewise linear complex with the following extra restriction.

Denoting the triangular sub-surface as ψ_i, we can write:

$$\psi_i = \bigcup_{i=1}^{i=n} \delta_i \tag{3.5}$$

where δ_i's are the faces of the carved-out sub-volume. Given an edge segment λ shared by two triangular facets δ_m and δ_n, then A must be empty, where A is set of all faces containing λ and:

$$A = \{\delta_i \mid \lambda \subset \delta_i, i \neq m, n\} \tag{3.6}$$

Using the same notation we call ψ_i non-closed if $n(B) = 1$ where

$$B = \{\delta_i \mid \Lambda \subset \delta_i, 1 \leq i \leq n\} \tag{3.7}$$

where Λ is an edge segment. This implies that ψ_i, i.e. the carved out volume, is a boundary separating only two different regions and none of its triangular edges can

be shared by more than two triangular faces.

The perimeter of a possible hole on surface of ψ_i can also contain a set of scattered nodes depending upon the extent of tetrahedral element deletion. Denoting all these simplexes as K, we cover the hole using edges and points from K. First M_i is recovered forming the original surface triangle. The algorithm also searches for other possible missing faces that can be reproduced using the simplexes available in K, removing them from K upon a successful recovery. Once the cavity is capped any remaining nodes (isolated during the carve out process) become part of the set of simplexes to be sent to LAST RESORT algorithm.

At this stage if the hole is completely covered, the sealed cavity is sent to our LAST RESORT algorithm to be meshed. Otherwise the remaining entities in K are examined to create new boundary faces in order to *stitch up* the hole. Figures 3.7 to 3.11 show an example of recovering a boundary face on a brain mesh. A pseudo code for recovery of the original surface boundary faces is provided in Algorithm 3.2.

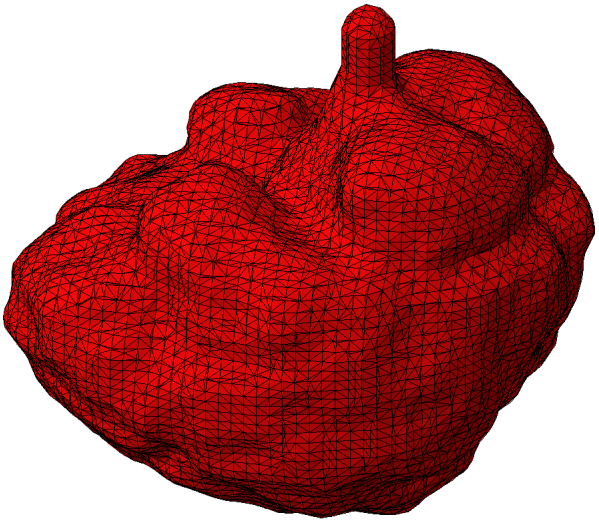


Figure 3.7: A tetrahedral mesh of an inverted human brain generated using 3D Delaunay algorithm.

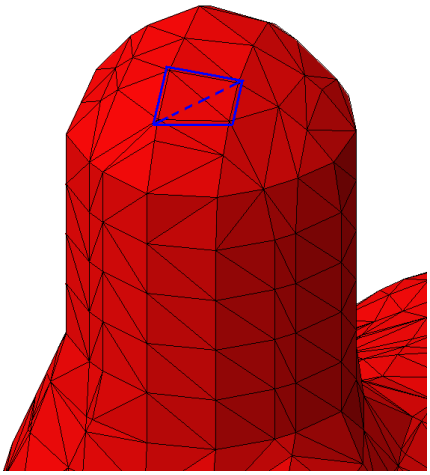


Figure 3.8: Detail of mesh where two of the original input boundary faces were missing (shown as blue lines).

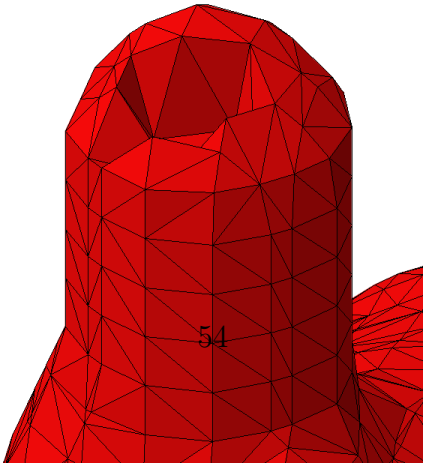


Figure 3.9: All the tetrahedrons containing one of the missing faces vertices are removed from mesh.

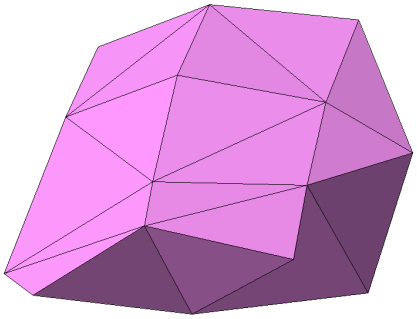


Figure 3.10: The carved out volume interface with mesh along with restored missing faces are used to seal the cavity in the mesh. This creates a closed surface suitable for LAST RESORT algorithm.

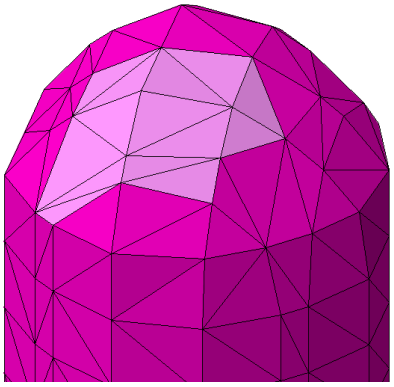


Figure 3.11: The output of LAST RESORT is inserted back to the original mesh with no inconsistencies since LAST RESORT preserves the cavity shell with fidelity.

The use of LAST RESORT on the closed surface ψ_i requires providing a desired quality value. A wide range of quality measures for tetrahedral elements has been devised. We chose to use the volume ratio quality, which measures the ratio of a given tetrahedron to that of a unilateral one.

If an acceptable quality sub-mesh cannot be created via the LAST RESORT algorithm, the failure is overwhelmingly related to a flawed or poor quality segment of the physical boundary facets. Introducing new internal nodes would rarely rectify this

situation containing flawed boundary facets. The LAST RESORT algorithm is exhaustive. Consequently, every possible connection, independent of Delaunay, is examined given the existing node deployment. If a satisfactory sub mesh cannot be created the user is notified of the root cause of problem so corrective measures can be taken. It should be noted that we have not encountered cases similar to **(author?)** [89] where the quality of facets of polyhedron is acceptable but no Delaunay tessellation was possible due to topology of boundary.

3.4 Results & Discussion

This post-processing method has been used on volume meshes generated with different methods: SPMESH [121], offset normal method [54] and Tetgen[104]. Our application is associated with alternative breast imaging strategies [80]. Meshes in Figure 3.12 and Figure 3.13 show a brain and a breast mesh. The breast tissue in Figure 3.13 is deformed due to the various imaging probes being applied to the tissue [80].

Test case 1 included the brain mesh which had 17,312 triangles and 8,658 nodes in the original physical boundary surface. 6,022 surface triangles were not present in the final volume mesh of 142,167 tetrahedrons. The parent mesh and surface mesh were sent to the LAST RESORT boundary recovery post processor. All missing surface facets were recovered such that the 17,312 original surface triangles were part of the final volume mesh. The updated volume mesh was subsequently sent to the LAST RESORT quality improvement post-processor. The overall quality was improved a small amount $q_{avg} = 0.410 \longrightarrow q_{avg} = 0.421$, however more importantly 1621 elements

whose qualities were virtually unacceptable were rectified. (The quality measure used here was a volume ratio similar to [63]). We should mention that the quality of input original mesh was not optimum and 817 faces out of 17312 had an area quality $q_{area} < 0.1$ which affects the improvement results.

In test case 2, a breast mesh was used containing 4,824 original boundary facets. The volume mesh was created using Tetgen containing 37,486 tetrahedral elements after removing the convex hull. The volume mesh was missing 579 original boundary faces which were all recovered after the mesh was post-processed. After returning from the LAST RESORT quality improvement post-processor the average quality was increased from $q_{avg} = 0.416$ to $q_{avg} = 0.427$. All 60 sliver and void elements were removed.

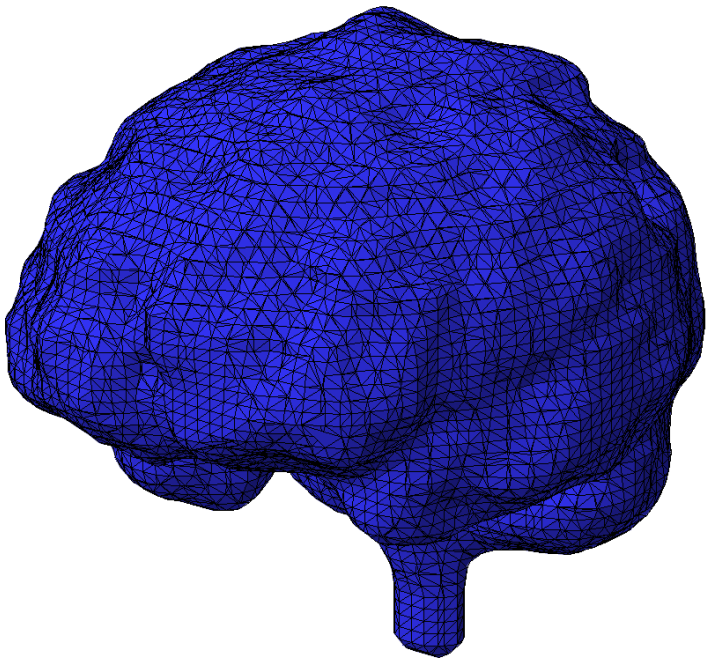


Figure 3.12: A volume mesh of a human brain.

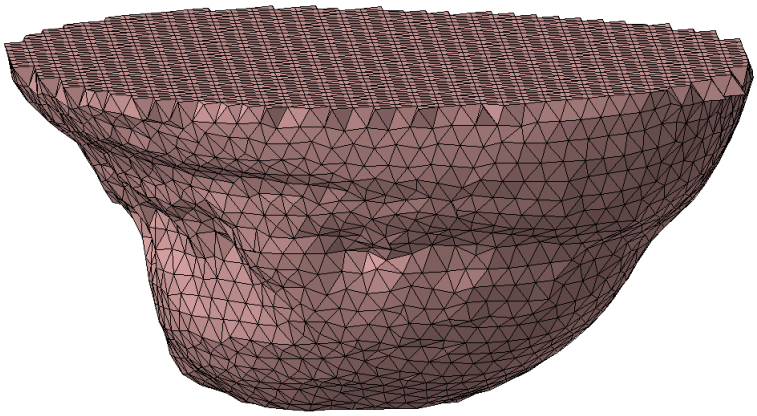


Figure 3.13: A volume mesh of a human breast.

3.4.1 Timing & comparison

We used 3 different implementations of Delaunay tetrahedralization algorithm for the mentioned test cases: *tetgen* [104], *qhull* [1] and our in-house 3D Delaunay. All of produced meshes using different methods contained almost the same number of missing faces and sliver elements. As described before, boundary recovery process tackles each missing face by removing adjacent tetrahedrons and performing a local topology transformation. Very often, remeshing of the cavity from a single missing face results in recovering multiple other missing faces. This contributes to performance increase of the algorithm. For example for the brain mesh with 17,312 missing faces, the LAST RESORT subroutine was only called 686 times. For the breast mesh this was 189 for 579 missing faces. The timing required to recover each missing face greatly depends on the size of cavity associated with it but in practice it averages between 0.15 seconds and 0.20 seconds. Table 3.1 summarizes the timings as well as number of LAST RESORT calls in order to re-mesh the cavity. Also using the breast mesh, we increased the resolution of the mesh to obtain finer meshes of the same geometry. Figure 3.14 shows how the number of missing faces varies based on the number of solid elements present in the mesh.

	# of elements	# of missing faces	# of LAST RESORT calls	avg. time
Breast 1	30,011	260	114	0.15 sec
Breast 2	50,310	282	124	0.20 sec
Brain	95,581	1,694	686	0.15 sec

Table 3.1: Performance of LAST RESORT in boundary recovery application.

Using LAST RESORT to enhance the quality of mesh and remove sliver elements

	# of slivers	# of LAST RESORT calls	avg. time for each sliver
Breast	148	104	1.08
Brain	310	241	1.22

Table 3.2: Performance of LAST RESORT in sliver removal application

required slightly different number of calls to LAST RESORT function. The brain mesh contained 310 sliver elements and LAST RESORT was called 241 times with 1.22 seconds spent on each try. In order to remove 148 sliver elements of breast mesh number of calls was 104 and average time spent to recover sliver tetrahedrons was 1.08 seconds (see Table 3.2).

We also compared our algorithm to *tetgen,* a 3D mesh generator freely available to public. 'tetgen' is based on Delaunay refinement algorithm proposed by Shewchuk [93]. 'tetgen' can create constrained Delaunay meshes with variable sizing and it attempts to remove sliver elements. The grading of mesh is done based on volume size of a tetrahedron rather than edge lengths and also it can not remove *all* sliver and compromised elements. The methods proposed so far to generate a mesh and perform post processing steps might not be able to achieve timings similar to that of 'tetgen' but it surpasses SPMESH. The excellent quality of elements inside the domain and virtually no sliver element close to boundary can make up for the lack of speed in our algorithms when it is compared against 'tetgen'. After all it is always preferable to spend more time preparing a good mesh for a finite element model than spending that time to evaluate and re-produce results that are based on a mesh with compromised elements.

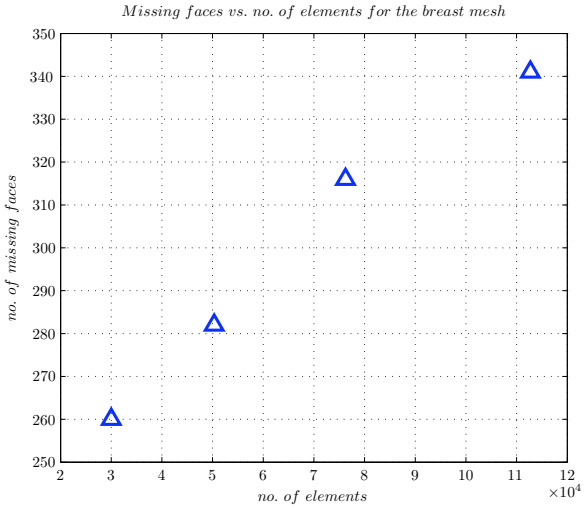


Figure 3.14: Number of missing faces versus the number of tetrahedral elements in the mesh.

Algorithm 3.1 Last Resort Flowchart

1: Last_Resort_Algorithm(Ψ_i,η_e)
2: $\Psi_i \leftarrow$ Boundary surface
3: $\eta_e \leftarrow$ Minimum desired element quality
4: Output: $PotentialMesh_j$
5: **if** Ψ_i has only 4 remaining faces **then**
6: Form the tetrahedral element e
7: $PotentialMesh_j \leftarrow e$
8: **return** $PotentialMesh_j$
9: **end if**
10: **for all** Faces, F_i, in current boundary surface Ψ_i **do**
11: **if** F_i can produce at least one acceptable tetrahedron whose quality $q > \eta_e$ **then**
12: $N_i \leftarrow$ number of possible tetrahedrons
13: **if** $N_i < N_{min}$ **then**
14: $N_{min} \leftarrow N_i$
15: Insert F_i to the top of $List_{faces}$ stack and push all the current faces down
16: **end if**
17: **else**
18: **return** NULL
19: **end if**
20: **end for**
21: **while** $List_{faces} \neq \emptyset$ **do**
22: Pick the face from top of the $List_{faces}$ stack and form its best tetrahedron e
23: Update Ψ_i boundary surface based on insertion of element e
24: Assign the new boundary to ψ_i
25: Call $M \leftarrow$ Last_Resort_Algorithm(ψ_i,η_e)
26: **if** M is not NULL **then**
27: $PotentialMesh_j = PotentialMesh_j \cup M$
28: **return** $PotentialMesh_j$
29: **else**
30: Remove the top enitity in $List_{faces}$ stack and continue
31: **end if**
32: **end while**
33: **return** NULL

Algorithm 3.2 Boundary Recovery

BoundaryRecovery(Γ,Θ)
Γ = Tetrahedral mesh
Θ = Original/Desired Boundary mesh
Output: Constrained Delaunay of Γ (an almost Delaunay mesh)
$\Omega \leftarrow$ Extract numerical boundary faces of Γ by considering faces that only belong to one tetrahedron
$M \leftarrow \Theta \cap \Omega'$ {Find list of missing faces}
for all Faces, M_i, in missing faces list M **do**
  $e \leftarrow$Find all tetrahedral elements that share at least one of the vertices of M_i
  Concatenate all members of e, leaving only their exterior surface $\rightarrow \psi$
  if ψ is non-closed surface **then**
    $K \leftarrow$ All the vertices and edges of the hole
    Construct missing face M_i as well as any additional missing faces in neighbourhood using members of K. Remove all the members of K used in construction of missing faces.
    if $K \neq \emptyset$ **then**
      Cover the hole by creating new numerical faces by using simplexes of K
    end if
  end if
  $Mesh_i$ = Call Last_Resort_Algorithm(Ψ_i,η_e)
  if Last Resort was successful **then**
    Insert $Mesh_i$ back to Γ
  else
    Move M_i to the bottom of missing face list M
  end if
  if $M = \emptyset$ or we have tried all missing faces in M **then**
    break the loop
  end if
end for
Return Modified Γ

Chapter 4

Mesh Smoothing

4.1 Introduction

The quality of elements can affect results and performance of discretization methods which are used to solve partial differential equations over a domain[32, 99]. These methods partition the domain into elements of simple geometrical form in order to find a solution function in the space of problem which approximates the actual solution to the differential equation [12]. Schewchuk [99] shows how the element quality can affect both discretization errors as well as stiffness matrix condition numbers in a finite element application. Additionally, the time cost of FEM solvers can be reduced by providing them with a better and optimized mesh [37]. This quality issue has prompted extensive study on optimization of mesh and its effects on implementation methods.

4.2 Related Work

Post processing of a mesh to optimize its overall quality and filter out compromised elements is almost ubiquitous among automatic mesh generators, especially three dimensional mesh generators since tetrahedral elements have more possibilities to be ill-formed than their 2D counterparts [34]. Classically, mesh improvement techniques could be categorized into 3 general areas, or their combinations:

1. Topological optimizations
2. Vertex set alteration (insertion or deletion of nodes)
3. Node displacement (geometric optimizations).

In topological optimization, connectivity of adjacent elements is altered to achieve a higher quality set of elements without changing the coordinates of vertices. This strategy can include diagonal swaps or face/edge swaps [53, 120, 63, 33]. The advantage of this approach is its easy implementation and effectiveness. We also recently proposed a robust, successful generalization of local transformation based on a LAST RESORT algorithm [44].

In second method, i.e. *insertion/deletion of vertices*, new vertices are introduced or removed from the mesh in order to improve the quality. When inserting, a point is placed in a strategically computed location to improve the local elements. This point, known as *Steiner* point, is usually centered in geometric locations such as the circumcenter of a compromised element [18, 17, 87]. Subsequently, this method was improved by work in [98] and [109]. Other researches have tried different methods to determine the insertion location [28].

In node displacement method or geometric optimization, the actual location of nodes are changed to achieve a better state of element quality, such as shape measures, or to optimize an *"energy field"* which is correlated with good quality elements. The measures to optimize can either be determined beforehand or after the solution is obtained to minimize the approximation errors.

The most popular method in this category is the Laplacian method and its variants in which a node is moved to the centroid of its immediate neighbours [3, 30, 62, 119]. Laplacian method is simple to implement and it runs fast but it does not necessarily improve the quality of elements. Also in this category, researchers try to minimize a given distortion metric as opposed to heuristic-based Laplacian algorithms[2, 5]. Knupp [56] proposes a method based on the optimization of Jacobian matrix norm. Oddy et. al. [77] used elasticity concepts for smoothing process. Freitag [35] and Canann et. al [15] combined the Laplacian method with optimization based approaches to achieve better results and faster timings. Furthermore, smoothing techniques based on simulation of repulsion and attraction forces between vertices has been studied [11, 70, 81, 103].

Majority of these methods perform their smoothing or optimization operation on a local basis and then iteratively visit every node in the mesh in order to obtain global improvement. More recently, global approaches have been introduced in which a system of equations is established for all the vertices, and solutions are obtained afterwards to determine the new location of nodes. In general, these methods define an objective function and its constraints and then apply a numerical solver to obtain the results. For example [75] chooses "inverse mean ratio metric" of a tetrahedral ele-

ment and minimizes it globally over the entire domain. Careful selection of objective function, constraining conditions and numerical solver are important for quantifiable results. This method can require a long time to run. Freitag et. al. and the reference within illuminate more details and constraints of this approach[36].

4.3 Proposed Method

One of the desired features of a Delaunay mesh in 2D is the fact that it maximizes the minimum angles of the triangle mesh [58]. This feature produces an optimum topology for a given set of vertices but, unfortunately, it does not extend to three dimensional tetrahedral meshes generated by Delaunay method. In fact, with nearly optimally spaced nodes the 3D Delaunay can yield unacceptable tetrahedral elements [52]. It has been shown that maximizing the solid angle of a Delaunay tetrahedral mesh can result in a mesh better than the original mesh [52].

Inspired by this property, we propose a method that repositions the nodes of the mesh to locally maximize the solid angles of the mesh. Solid angle subtended by the vertex v of a tetrahedron is proportional to the spherical area of the projection of opposing face to v on a unit sphere centered at v. In general a solid angle can be defined as (Figure 4.1:

$$\Omega = \iint_S \frac{\vec{r}.\hat{n}}{r^3} dS \tag{4.1}$$

where $\vec{r}$ is the vector from origin to surface patch dS and $\hat{n}$ is unit vector normal to patch. For a regular tetrahedron (equal side lengths) all 4 solid angles are equal

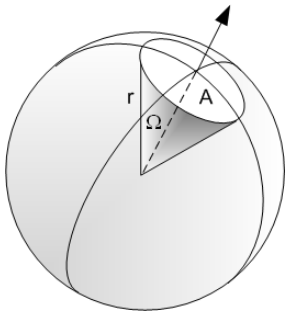


Figure 4.1: A Solid angle Ω.

to:

$$3.\arccos(1/3) - \pi \backsimeq 0.551\ str \tag{4.2}$$

The interior nodes of the mesh and the attached elements are examined to estimate the gradient of solid angle with respect to current vertex. All the adjacent nodes of the current vertex are kept fixed while the vertex is moved in the gradient direction. The node is relocated continuously in the optimum direction till no further local solid angle increase is attainable. This process iterates over all the mesh nodes until a preset mesh quality is achieved or the global improvement converges. User can choose to keep the boundary nodes completely unaltered or they can be repositioned in a tangential motion with respect to the original boundary facets.

4.3.1 Algorithm

The process starts with identifying low quality elements in the mesh. An interior node is selected as a *free* node and all tetrahedrons connected to it are extracted, forming a cluster of elements. All the nodes within the cluster are considered fixed

except for the free vertex. The goal is to increase the minimum solid angle within this mesh cluster by relocating the free node. To achieve this, algorithm needs to calculate a gradient for the solid angle change, i.e., determine the gradient direction of the node to attain the highest increase in the minimum solid angle of the cluster.

Let $\xi_i(x) \mid_{1 \leqslant i \leqslant n}$ represent the value of n solid angles in the cluster which are a function of current location x of the free node. Our goal is to find a displacement vector $\vec{\Delta}(x)$ that will determine new location of free node yielding the highest increase in the current minimum solid angle:

$$\vec{X}_{new} = \vec{X}_{old} + \vec{\Delta}(x) \tag{4.3}$$

where $\vec{X}$ is the location of free node. Using a linear approximation for $\xi_i(x)$ functions, one can obtain the new solid angle values if the free point was moved by $\vec{d}(x) = (dx_1, dx_2, dx_3)$:

$$\xi_i^{m+1} = \xi_i^m + \frac{\partial \xi_i^m}{\partial x_1} dx_1 + \frac{\partial \xi_i^m}{\partial x_2} dx_2 + \frac{\partial \xi_i^m}{\partial x_3} dx_3 \tag{4.4}$$

To calculate the gradient of ξ_i, i.e.

$$\nabla \xi_i = \left(\frac{\partial \xi_i}{\partial x_1}, \frac{\partial \xi_i}{\partial x_2}, \frac{\partial \xi_i}{\partial x_3} \right) \tag{4.5}$$

we introduce an infinitesimal change in x_1 direction while constraining other two directions to zero. Then from Equation 4.4:

$$\frac{\partial \xi_i}{\partial x_1} = \frac{\xi_i^{m+1} - \xi_i^m}{dx_1} \tag{4.6}$$

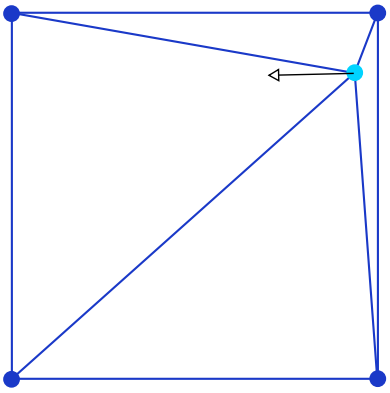
Repeating the same steps for other two spatial directions $\nabla \xi_i$ is obtained for every solid angle. Since the objective is maximizing the minimum angle, we only consider the gradient of $\Phi(x)$ at the current location of the free node in local mesh cluster:

$$\Phi(x) = \min_{1 \leqslant i \leqslant n} \xi_i(x) \tag{4.7}$$

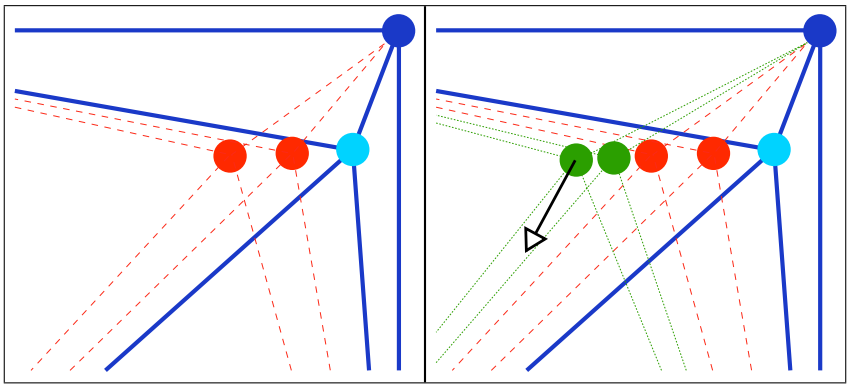
At this point the free node is moved by a distance δ which is dependent of cluster size in $\vec{\Delta}(x)$ direction. The size of δ is based on the cluster size to prevent over stepping its bounds or bypassing an optimal locations. The algorithm obtains displacement vector as:

$$\vec{\Delta}(x) = \delta \cdot \frac{\nabla \Phi(x)}{|\nabla \Phi(x)|} \tag{4.8}$$

and reposition the free node to its new location, Figure 4.2(a). The process keeps moving the free vertex in $\vec{\Delta}(x)$ direction until the minimum solid angle ceases to improve at which point the step size δ is reduced by a predefined factor, Figure 4.2(b) and (c). This is done since a finer step might put the free node closer to the local peak without skipping over it because of a large displacement step. If the finer δ would increase the minimum angle, the free node will be displaced in $\vec{\Delta}(x)$ direction. Otherwise, if resorting to a finer step fails to improve the solid angle the algorithm recalculates the $\vec{\Delta}(x)$ to obtain a new direction of repositioning. The above steps are iterated for the current free node until no further angle increase is obtained. Amenta et. al [2] show the locus of possible positions of the free node is convex provided a



(a)

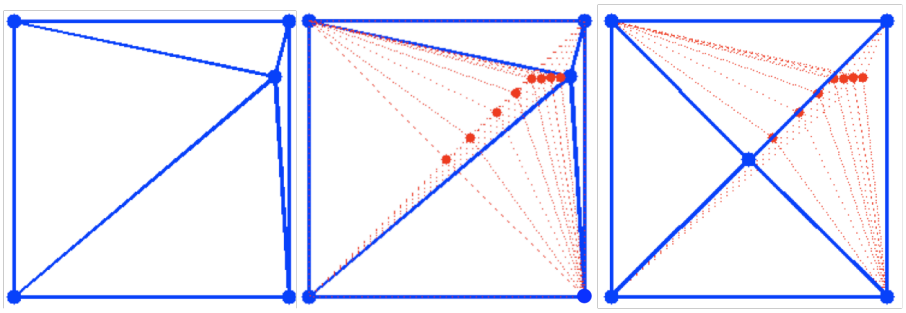


(b) (c)

Figure 4.2: (a) Initial gradient of free node. (b) *Closer look at free node:* Coarse steps in gradient direction taken. (c) *Closer look at free node:* After coarse steps fail, algorithm resorts to finer steps till no more improvement is possible.

bound on the quality of desired local cluster is given.

This process is iterated over all compromised elements of the mesh until the quality criterion is met or optimization converges. Figure 4.3 shows a 2D example of the smoothing algorithm. It can be seen how the gradient gets updated as moving the free node is being displaced from its original location.



(a) (b) (c)

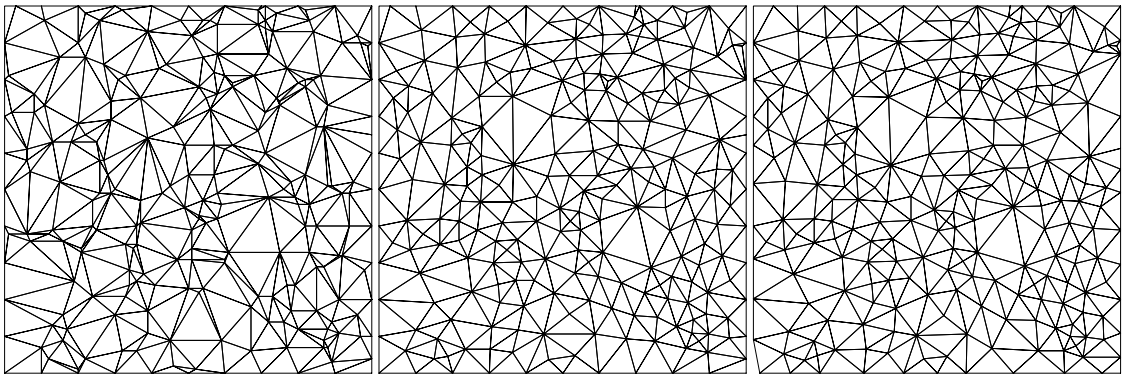
Figure 4.3: 2D Example of the algorithm. (a) Original mesh with one free node. (b) The nodes is moved along initial gradient in small steps. A new gradient is recalculated as soon as the current one seizes to improve the minimum angle. (c) Optimized location of the node.

4.4 Results & Discussions

To show a better demonstration of how the algorithm smooths the mesh, we first show a 2D example. The mesh in Figure 4.4 is a box with randomly generated points inside it. It consists of 234 nodes and 434 elements. Several measures are used to quantify the quality of mesh [31]. In this work we choose the aspect ratio quality of an element which is the ratio of volume (area in 2D) of element to sum of squares of edge lengths. For a tetrahedron we can calculate it as:

$$q = 12\frac{\sqrt[3]{(3V)^2}}{\sum_{i=1}^{6} L_i^2} \tag{4.9}$$

Also extremum of angles in addition to a histogram of all angles are provided. Figure 4.4(b) is the optimized mesh which was obtained after 20 iterations over the entire domain, not just the low quality elements with small angles. To create the mesh in Figure 4.4(b), all the boundary nodes were fixed while all the interior nodes



(a) (b) (c)

Figure 4.4: 2D Experimentation of the algorithm. (a) original mesh. (b) Optimized mesh with fixed boundary nodes. (c) Optimized mesh with sliding boundary nodes.

were considered as free nodes. On the other hand, Figure 4.4(c) shows the same mesh but this time we let the algorithm slide the boundary nodes tangentially. Reader can notice the biggest visual difference at the bottom and top right corner of the optimized meshes. One can easily implement a feature to keep 4 corners of the mesh fixed while allowing other boundary nodes to move tangentially in order to avoid the smoothed lower left corner in this example. As Table 4.1 shows, the latter mesh produces a better quality mesh in terms of angle distribution and minimum angle of the mesh. A full histogram of the original mesh and its two smoothed counterparts are shown in Figures 4.5 and 4.6. Note how an optimization done with sliding nodes has eliminated all the angles in $\alpha < 20^\circ$ and $\alpha > 120^\circ$ range.

We also experimented the algorithm on 3D meshes. Shown in Figure 4.7 is a brain mesh constructed using SPMESH [121]and surface data is courtesy of Dr. Miga

	q	$min\ \alpha$
original	0.683	3.54°
Mesh (a)	0.849	6.34°
Mesh (b)	0.854	20.05°

	$0-20°$	$20-60°$	$60-90°$	$90-120°$	$120-180°$
original	118	576	364	199	45
Mesh (a)	7	754	350	182	4
Mesh (b)	0	767	357	174	4

Table 4.1: Angle histogram of the 2D meshes of Figure 4.4

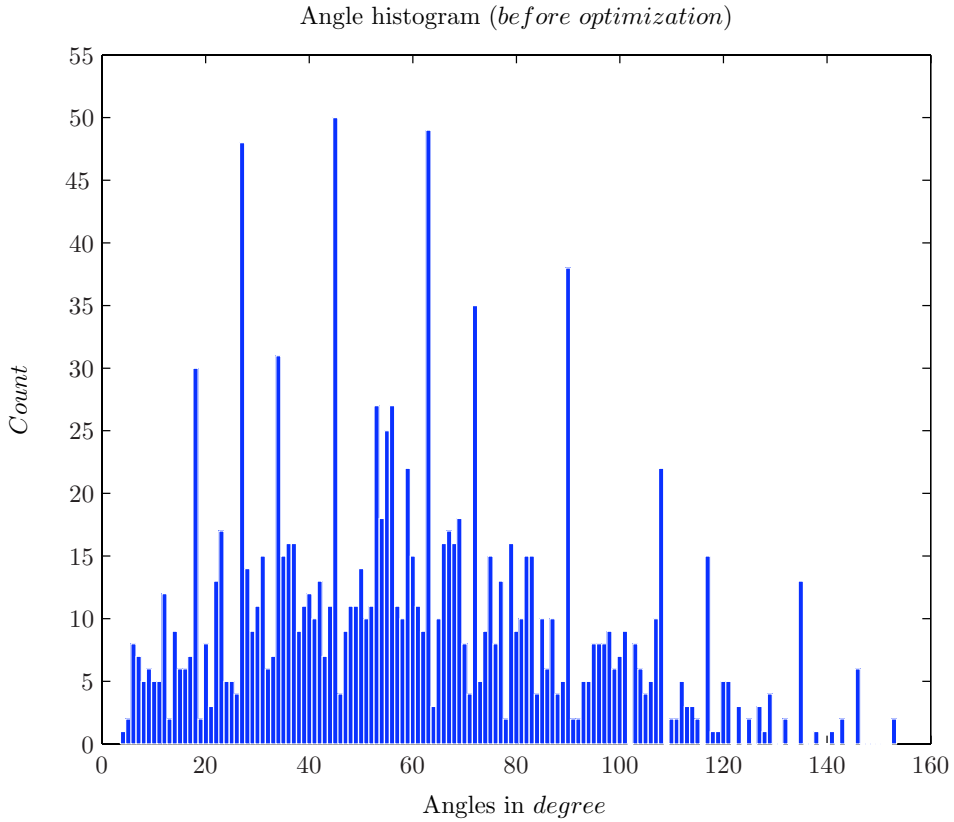
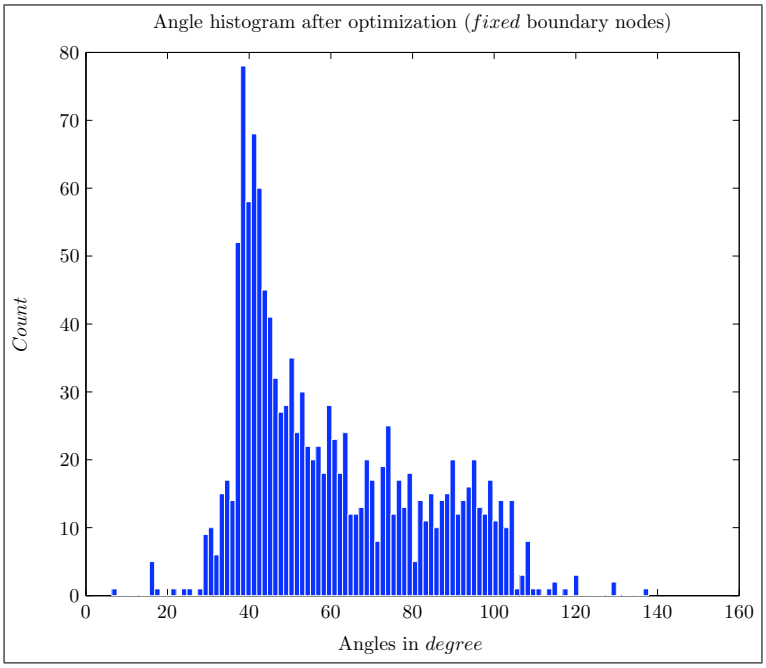
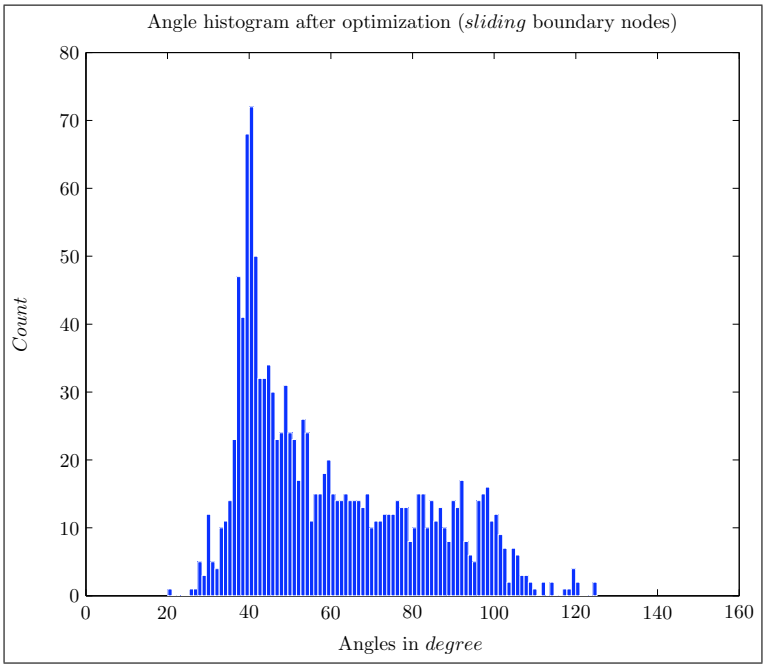


Figure 4.5: Angle histogram of the 2D mesh in Figure 4.4 before optimization.



(a) Fixed boundary nodes



(b) Sliding boundary nodes

Figure 4.6: Angle histogram after optimization.(a) Optimization with fixed boundary nodes.(b) Optimization with sliding boundary nodes.

[71]. The mesh contains 39,876 elements and 7,768 nodes. To measure the quality improvement of the mesh, we measured dihedral angles of tetrahedrons as well as the solid mesh. Figure 4.8 shows the histogram of dihedral[1] angles before and after optimization while Figure 4.9 is portion of solid angle histogram. Note how the histogram is shifted toward the higher solid angles ($> 0.2\,stradians$). The optimization was done using a *sliding* boundary method. For this mesh convergence criterion is based on node movement, i.e. the algorithm will terminate when improving the solid angles requires node displacements much smaller than a preset value based on the minimum edge size of the mesh. Figure 4.11 shows the convergence rate before and after each iterations. Note that the first 2 to 3 iterations include the steepest increase in the solid angle and the consequent iterations can be considered as fine tuning the location of the vertices to obtain the optimum position.

The average quality of mesh using Equation 4.9 was increased to 0.8726 up from 0.8641 while the minimum quality for this particular mesh was at a constant value of 0.2296. Figure 4.10 shows a linear relation for time requirements for optimization and the number of elements.

[1]Dihedral angles can be related to solid angles using $\sum_{\{i=1\}}^{4} \Omega_i = 2\sum_{i=1}^{6} \phi_i - 4\pi$ where ϕ_i are the dihedral angles.

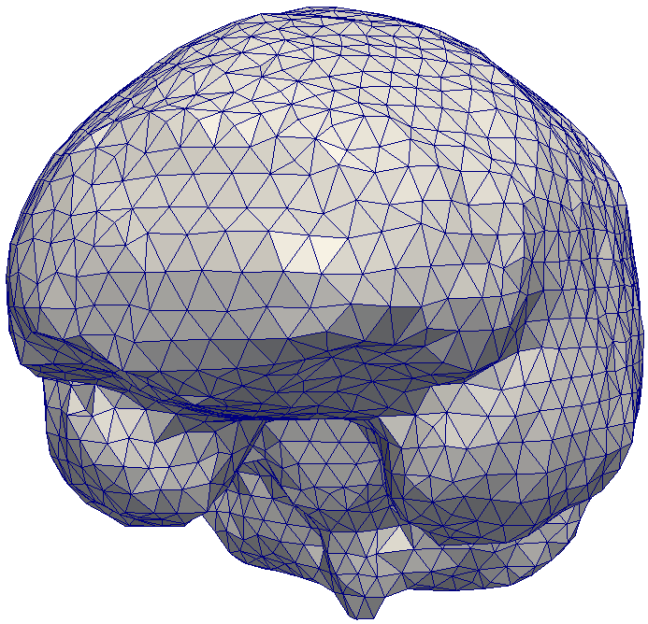


Figure 4.7: A brain mesh generated from reconstructed surfaces off of 2D images.

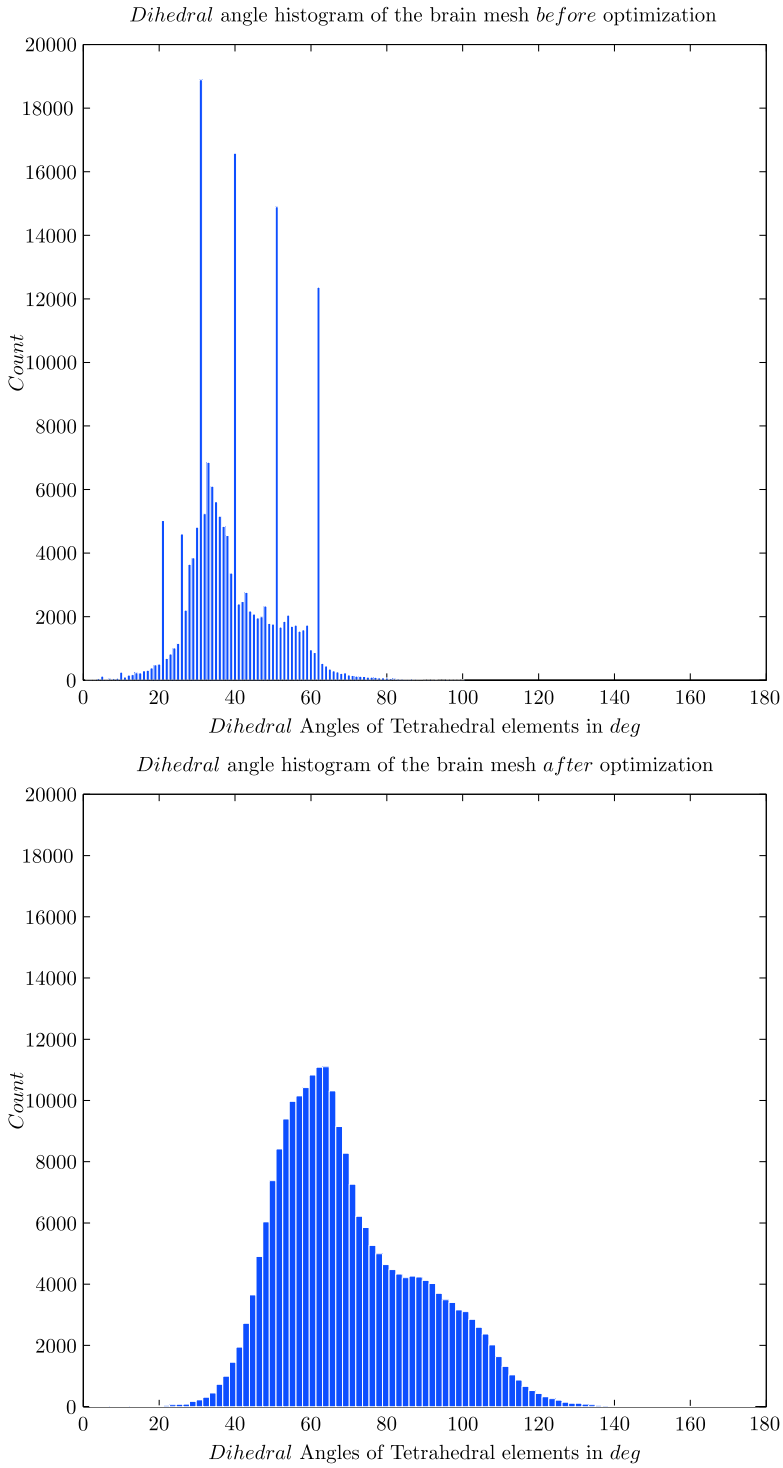


Figure 4.8: Histogram of dihedral angle of the brain mesh.

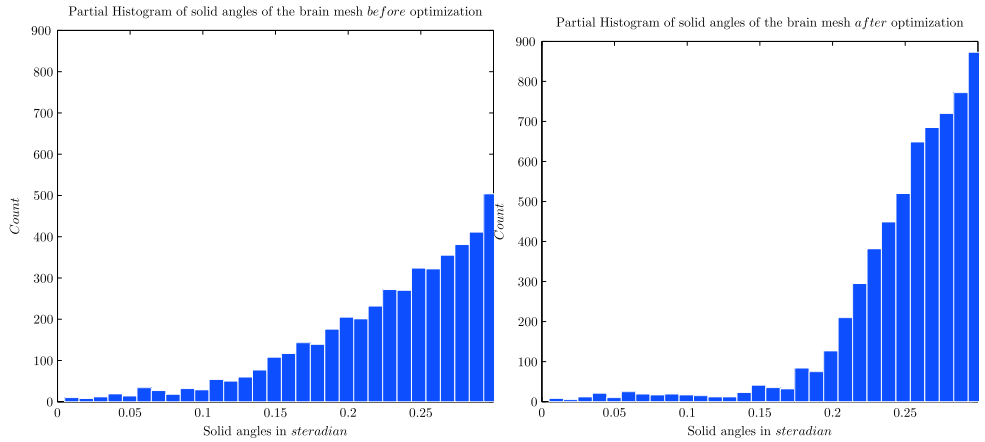


Figure 4.9: Partial histogram of solid angles obtained from the brain mesh.

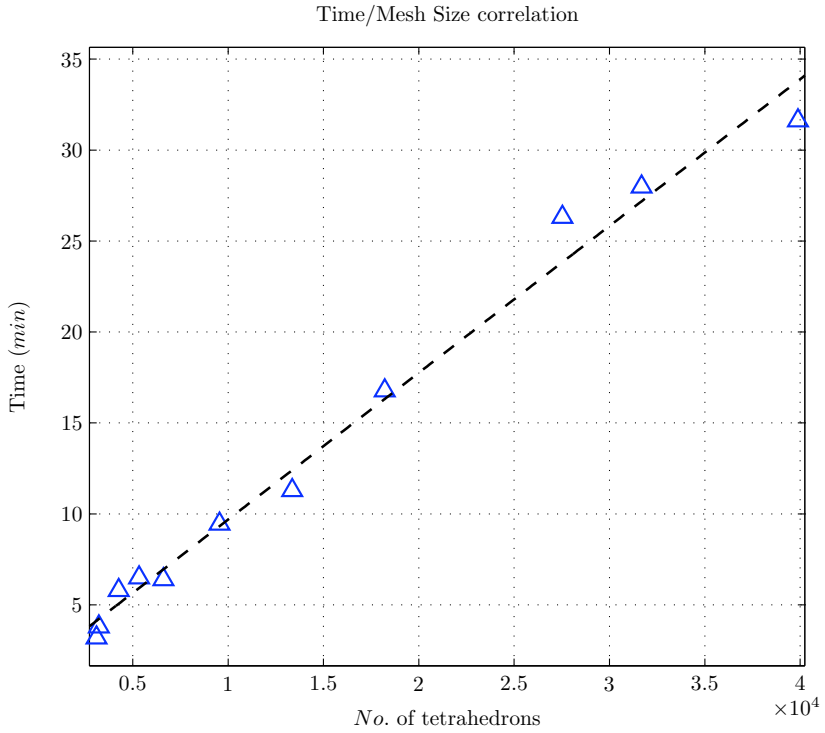


Figure 4.10: Linear time requirements in terms of number of tetrahedral elements in the mesh.

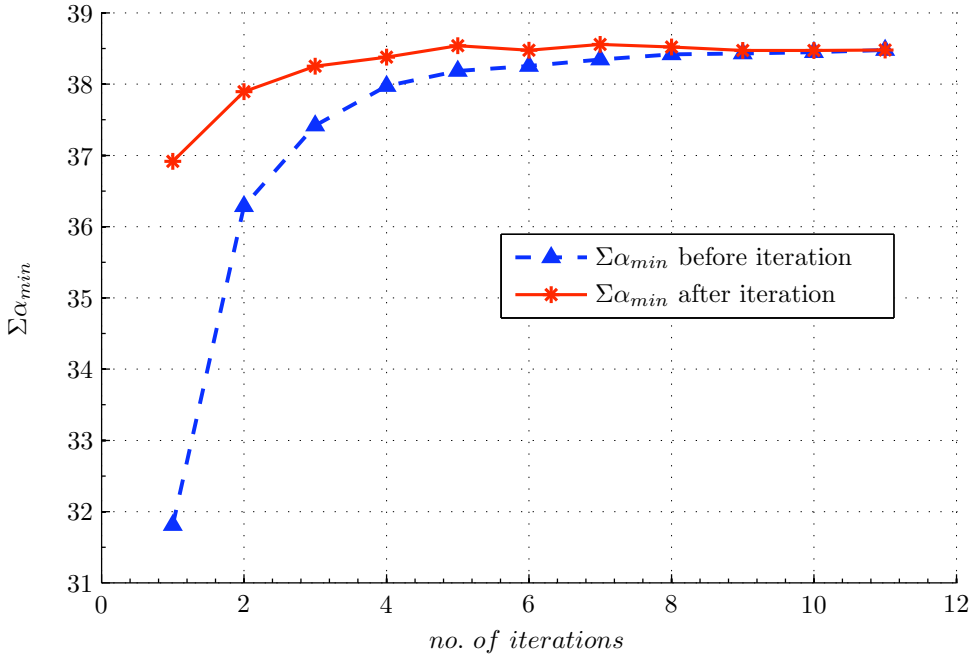


Figure 4.11: Number of iterations required to optimize the brain mesh.

Chapter 5

Conclusions

In this dissertation we tackled three major problematic areas in modern automatic mesh generation: node deployment and element grading, boundary recovery and element quality as well as post processing and mesh smoothing.

We presented a template based grid generation coupled with Delaunay tetrahedralization that provides great flexibility in mesh size and density variation. Using a triangulated surface mesh to define the domain, a 3D matrix template is overlaid to enclose the bounding box. The template is then exclusively used to guide the algorithm through the mesh and to decide spatial location of the grid points to be placed based on user-defined density requirements. Since the matrix indices are used to locate points, measure distances and adjust edges sizes, the algorithm enjoys a fast and robust venue to explore the mesh. This is achieved by initially tagging pixels of template closest to input boundary nodes and a buffer zone of pixels is established around each node whose extent is spatially dependent on density requirements. Moving in-

ward the domain, the pixels of template, which are not processed yet, are examined for possible placement of a new node. Each new node and its neighboring pixels are marked as *processed.* The template not only helps avoid floating point calculations and round-off errors but also it eliminates the need to process working surfaces or partial meshes. Although the quality of input surface will dictate the shape of tetrahedral elements adjacent to it, our algorithm is fairly insensitive to shape of surface triangles in order to create good elements.

A post-processing tool was presented that recovers the original surface boundary for both interior and exterior boundaries using local transformations in connectivity rather than extra vertex insertions. This ability was generally lacking in the field, yet of critical importance for remeshing a sub-zone of a large parent mesh or replacement of a material region. Our proposed method can handle any type of volume mesh generated by various methods and recover the initial physical boundary. To achieve this, we introduced LAST RESORT, an efficient search algorithm capable of creating quality grids using a closed polyhedron. Missing boundary faces of the given mesh are identified and their neighboring tetrahedral elements are removed creating an open cavity. The hole on the cavity, resulted from removal of the rogue faces, is covered using all possible physical boundary faces that were missing in this region. The sealed cavity is passed to LAST RESORT to create a sub-mesh that can be inserted directly into the parent mesh. We have been successfully using this approach in biomedical applications and alternative tissue imaging applications which use a variety of mesh generations systems.

Mesh optimization and smoothing are necessary for a reliable and fast FEM solu-

tion. Seeking the angle property of 2D Delaunay in higher dimension, we proposed an optimization algorithm that maximizes the minimum solid angle of tetrahedral elements. The smoothing process is performed on a local cluster of tetrahedral elements that are connected to a *free* node. Using a linear approximation method, we solve for the gradient direction. The free node is displaced using that direction and a small step size which is carefully chosen in order to keep the validity of cluster intact. After each step the algorithm checks the improvement. If no improvement is realized or a different vertex poses as the minimum solid angle, the gradient calculation is repeated to keep the free node moving in the steepest increase direction until no further improvement is obtainable. This process is repeated iteratively for all free nodes of the mesh. The algorithm optimizes the mesh regardless of element generation method. Our method easily improves the element with low quality or compromised shape.

Bibliography

[1] Qhull. http://www.qhull.org/.

[2] Nina Amenta, Marshall Bern, and David Eppstein. Optimal point placement for mesh smoothing. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 528–537, 1997.

[3] E Amezua, M. V. Hormaza, A. Hernandez, and M. B. G. Ajuria. A method for the improvement of 3D solid finite-element meshes. *Advances in Engineering Software*, 22(1):45–53, 1995.

[4] T.J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5(3-4):161–75, Summer-Fall 1989.

[5] R. E. Bank and R. K. Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, JUN 1997.

[6] R.E. Bank and A.H. Sherman. Refinement algorithms and data structures for regular local mesh refinement. *Scientific Computing*, pages 3–17, 1983.

[7] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. *Discrete & Computational Geometry*, 14(4):411–428, Dec 1995.

[8] Ted D. Blacker and Myers R.J. Seams and wedges in plastering: A 3d hexahedral mesh generation algorithm. *Engineering with Computers*, 2:83–93, 1993.

[9] H. Borouchaki, P.L. George, and S.H. Lo. Optimal Delaunay Point Insertion. *International Journal for Numerical Methods in Engineering*, 39(20):3407–3437, Oct 1996.

[10] H. Borouchaki and S.H. Lo. Fast Delaunay triangulation in three dimensions. *Computer Methods in Applied Mechanics and Engineering*, 128(1-2):153–167, Dec 1995.

[11] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *5th International Meshing Roundtable*, pages 63–76, 1996.

[12] S. C. Brenner and L. R. Scorr. *The mathematical theory of finite element methods*. Springer, Berlin Heidelberg, New York, 2002.

[13] M. Bro-Nielsen. Finite element modeling in surgery simulation. *Proceedings of the IEEEE*, 86(3):490–503, Mar 1998.

[14] Scott A. Canann. *Plastering and Optismoothing: New Approaches to Automated, 3D Hexahedral Mesh Generation*. PhD thesis, Brigham Young University, 1991.

[15] Scott A. Canann, Joseph R. Tristano, and Matthew L. Staten. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. In *Proceedings of the 7th International Meshing Roundtable*, pages 479–494, 1998.

[16] Center for Morphometric Analysis - Massachusetts General Hospital. Internet Brain Segmentation Repository. http://www.cma.mgh.harvard.edu/ibsr/.

[17] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280, New York, NY, USA, 1993. ACM.

[18] L. Paul Chew. Guaranteed-quality delaunay meshing in 3d. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 391–393, New York, NY, USA, 1997. ACM.

[19] L.P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989.

[20] D. Cohen-Steiner, E.C. Verdiere, and M. Yvinec. Conforming Delaunay triangulations in 3D. *Computational Geometry-Theory and Applications*, 28(2-3):217–233, Jun 2004.

[21] H.H. Dannelongue and P.A. Tanguy. 3-dimensional adaptive finte-element computations and applications to non-newtonian fluids. *International Journal for Numerical Methods in Fluids*, 13(2), 1991.

[22] E.F. D'Azevedo and R.B. Simpson. On optimal interpolation triangle incidences. *SIAM Journal on Scientific and Statistical Computing*, 10:1063–1075, 1989.

[23] B.N. Delaunay. Sur la sphere. *Vide. Izvestia Akademia Nauk SSSR*, 7:793–800, 1934.

[24] Tamal K. Dey, Chanderjit L. Bajaj, and Kokicki Sugihara. On good triangulations in three dimensions. In *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 431–441, New York, NY, USA, 1991. ACM.

[25] T.K. Dey, C.L. Bajaj, and K. Sugihara. On good triangulations in three dimensions. *Int. J. Comput. Geom. Appl.*, 2(1):75 – 95, 1992.

[26] Q. Du and D. Wang. Constrained boundary recovery for three dimensional Delaunay triangulations. *International Journal for Numerical Methods in Engineering*, 61:1471–1500, 2004.

[27] H. Edelsbrunner, X.Y. Li, G. Miller, A. Stathopoulos, D. Talmor, S.H. Teng, A. Ungor, and N. Walking. Smoothing and cleaning up slivers. In *Proceedings of 32nd Annual ACM Symposium on Theory of Computing*, pages 273–277, May 2000.

[28] Herbert Edelsbrunner and Damrong Guoy. Sink insertion for mesh improvement. *International Journal of Foundations of Computer Science*, 13(2):p223, 04 2002.

[29] D. Eppstein. Faster circle packing with application to nonobtuse triangulation. *Int. J. Comput. Geom. Appl. (Singapore)*, 7(5):485 – 91, 1997.

[30] D. A. Field. Laplacian smoothing and Delaunay triangulations. *Communications in Applied Numerical Methods*, 4(6):709–712, Nov-Dec 1988.

[31] D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47(4):887–906, Feb 2000.

[32] I. Fired. Condition of finite element matrices generated from nonuniform meshes. *AIAA*, 10:219–221, 1972.

[33] L. A. Freitag and C. OllivierGooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, Nov 1997.

[34] L.A. Freitag and C. Ollivier-gooch. A comparison of tetrahedral mesh improvement techniques. In *Fifth International Meshing Roundtable*, 1996.

[35] Lori A. Freitag. On combining Laplacian and optimization-based mesh smoothing techniques. In *AMD-Vol. 220: Trends in Unstructured Mesh Generation, ASME*, pages 37–43, July 1997.

[36] Lori A. Freitag, Patrick M. Knupp, Todd Munson, and S. Shontz. A comparison of optimization software for mesh shape-quality improvement problems. In *Proceedings of 11th International Meshing Roundtable*, Aug 2002.

[37] Lori A. Freitag and C Ollivier-Gooch. A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency. *International Journal of Computational Geometry & Applications*, 10(4):361–382, Aug 2000.

[38] P.J. Frey, H. Borouchaki, and P.L. George. 3d delaunay mesh generation coupled with an advancing-front approach. *Computer Methods In Applied Mechanics and Engineering*, 157(1-2):115–131, APR 28 1998.

[39] W.H. Frey. Selective refinement: A new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11):2183–2200, 1987.

[40] M. Galli, J. Botsis, and J. Janczak-Rusch. An elastoplastic three-dimensional homogenization model for particle reinforced composites. *Computational Materials Science*, 41(3):312–321, JAN 2008.

[41] P.L. George. Improvements on Delaunay-based three-dimensional automatic mesh generator. *Finite Elements in Analysis and Design*, 25(3-4):297–317, 1997.

[42] P.L. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements.* Hermes, Paris, 1998.

[43] P.L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering*, 92(3):269 – 288, 1991.

[44] H.R. Ghadyani, J.M. Sullivan, and Ziji Wu. Boundary recovery for Delaunay

tetrahedral meshes using local topological transformations. *Finite Elements in Analysis and Design*, accepted for publication, 2009.

[45] T. Grogsges, H. Borouchaki, and D. Barchiesi. New adaptive mesh development for accurate near-field enhancement computation. *Journal of Microscopy*, 229(2):293–301, 2008.

[46] Rt Hart, Vv Hennebel, N. Thongpreda, Wc Vanbuskirk, and Rc Anderson. Modeling the biomechanics of the mandible - a 3-dimensional finite-element study. *Journal of Biomechanics*, 25(3):261–286, Mar 1992.

[47] C. Hazlewood. Approximating constrained tetrahedrizations. *Computer Aided Geometric Design*, 10:67–87, 1993.

[48] K. Ho-Le. Finite element mesh generation methods - A review and classification. *Computer-Aided Design*, 20(1):27–38, Jan-Feb 1988.

[49] C.M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer*, 22(3):31 – 9, 1989.

[50] Y. Ito, P. C. Shum, A. M. Shih, B. K. Soni, and K. Nakahashi. Robust generation of high-quality unstructured meshes on realistic biomedical geometry. *International Journal For Numerical Methods In Engineering*, 65(6):943–973, FEB 5 2006.

[51] Songbai Ji, Ziji Wu, A. Hartov, D. W. Roberts, and K.D. Paulsen. Mutual-information-based patient registration using intraoperative ultrasound in image-guided neurosurgery. *Medical Physics*, 35(10):4612–4624, 2008.

[52] Barry Joe. Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation. *International Journal for Numerical Methods in Engineering*, 31(5):987–997, 1991.

[53] Barry Joe. Construction of 3-Dimensional improved-quality triangulations using local transformations. *Siam Journal on Scientific Computing*, 16(6):1292–1307, Nov 1995.

[54] B.P. Johnston and J.M. Sullivan. A Normal Offsetting Technique For Automatic Mesh Generation in 3 Dimensions. *International Journal for Numerical Methods in Engineering*, 36(10):1717–1734, May 1993.

[55] Camille Jordan. *Cours d'Analyze de l'École Polytechnique*. 1887.

[56] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part II - A framework for volume mesh optimization and the condition number of the Jacobian matrix. *International Journal For Numerical Methods In Engineering*, 48(8):1165–1185, JUL 20 2000.

[57] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18(3):199–210, 2002.

[58] Charles L. Lawson. Software C^1 surface interpolation. In John R. Rice, editor, *Mathematical software III*, pages 161–194, New York, 1977. Academic Press.

[59] A. Liu and M. Baida. How far flipping can go towards 3D conform-

ing/constrained triangulation. In *9th International Meshing Roundtable*, pages 295–206, 2000.

[60] A.W. Liu and B. Joe. On the shape measure of tetrahedra from bisection. *Mathematics of Computation*, 63(207), 1994.

[61] J.F. Liu, B. Chen, and Y. Chen. Boundary recovery after 3D Delaunay tetrahedralization without adding extra nodes. *International Journal for Numerical Methods in Engineering*, 72(6):744–756, Nov 2007.

[62] S. H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21(8):1403–1426, 1985.

[63] S. H. Lo. Optimization of tetrahedral meshes based on element shape measures. *Computer and Structures*, 63(5):951–961, Jun 1997.

[64] S.H. Lo. Delaunay triangulation of non-convex planar domains. *International Journal for Numerical Methods in Engineering*, 28(11):2965–2707, 1989.

[65] S.H. Lo. Volume Discretization into Tetrahedra-I. Verification and Orientation of Boundary Surfaces. *Computer and Structures*, 39(5):493–500, 1991.

[66] S.H. Lo. Volume discretization into tetrahedra II. 3D triangulation by advancing front approach. *Computer and Structures*, 39(5):501–511, 1991.

[67] S.H. Lo and W.X. Wang. Generation of tetrahedral mesh of variable element size by sphere packing over an unbounded 3d domain. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49), 2005.

[68] R Lohner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12(3-4):186–210, 1996.

[69] R Lohner. A parallel advancing front grid generation scheme. *International Journal For Numerical Methods In Engineering*, 51(6):663–678, JUN 30 2001.

[70] R. Lohner, K. Morgan, and O. C. Zienkiewicz. Adaptive grid refinement for the compressible Euler equations. *Accuracy estimates and adaptive refinements in finite element computations*, pages 281–297, 1986.

[71] Michael I. Miga. Biomedical modeling laboratory. http://bmlweb.vuse.vanderbilt.edu/.

[72] G.L. Miller, D. Talmor, S.H. Teng, and N. Walkington. A Delaunay Based Numerical Method for Three Dimensions: Generation, Formulation and Partition. In *Annual ACM Symposium on the Theory of Computing*, pages 683–692, Las Vegas, 1995.

[73] S.A. Mitchell and S.A. Vavasis. Quality mesh generation in higher dimensions. *Siam Journal On Computing*, 29(4):1334–1370, MAR 6 2000.

[74] Tomas Moller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *Doktorsavhandlingar vid Chalmers Tekniska Hogskola*, (1425):109 – 115, 1998. Ray-triangle intersections;.

[75] Todd Munson. Optimizing the quality of mesh elements. Technical Report ANL/MCS-P1260-0605, Argonne National Laboratory, Argonne, IL, June 2005.

[76] Michael Murphy, David M. Mount, and Carl W. Gable. A Point-Placement Strategy for Conforming Delaunay Tetrahedralization. In *Proceedings of the Eleventh Annual Symposium on Discrete Algorithms*, pages 67–74. Association for Computing Machinery, 2000.

[77] A. Oddy, J. Goldak, M. McDill, and M. Bibby. A distortion metric for isoparametric finite elements. *Transactions of the Canadian Society For Mechanical Engineering*, 12(4):213–217, 1988.

[78] S.J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 239–267, 1998.

[79] N. Palle and J.A. Dantzig. An adaptive mesh refinement scheme for solidification problems. In *Metallurgical and Materials Transactions A - Physical Metallurgy and Materials Science*, volume 27, pages 707–717, Mar 1996.

[80] K.D. Paulsen, P.M. Meaney, and L.C. Gilman. *Alternative Breast Imaging: Four Model-based Approaches*. Springer, 2005.

[81] P. O. Persson and G. Strang. A simple mesh generator in Matlab. *SIAM review*, pages 329–346, 2004.

[82] V.T. Rajan. Optimality of the Delaunay Triangulation in $\mathbb{R}^d$. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 357–363, 1991.

[83] M.C. Rivara. Mesh refinement processes based on the generalized bisection of simplices. *SIAM Journal on Numerical Analysis*, 21(3), 1984.

[84] J Roach, S Sharma, M Kapustina, and CW Carter. Structure alignment via delaunay tetrahedralization. *Proteins-Structure Function and Bioinformatics*, 60(1):66–81, JUL 1 2005.

[85] I. Roch, P. Bidaud, D. Collard, and L. Buchaillot. Fabrication and characterization of an su-8 gripper actuated by a shape memory alloy thin film. *Journal of Micromechanics and Microengineering*, 13(2):330–336, Mar 2003.

[86] Jim Ruppert. New and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 83 – 92, New York, NY, USA, 1993.

[87] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, May 1995.

[88] R Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering With Computers*, 12(3-4):168–177, 1996.

[89] E. Schonhardt. Uber die Zerlegung von Dreieckspolyedern in Tetraeder. *Mathematische Annalen*, 98:309–312, 1928.

[90] W.P. Segars and A. Tsuji. Study of the efficacy of respiratory gating in myocardial SPECT using the new 4-D NCAT phantom. In *IEEE Trans Nucl Sci*, 2002.

[91] Mark S. Shephard and Marcel K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 32(4):709 – 749, 1991.

[92] J.R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18:305–363, 1997.

[93] J.R. Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the Fourteenth Annual Proceedings of Fourteenth Annual Symposium on Computational Geometry*, 1998.

[94] J.R. Shewchuk. Lecture Notes on Delaunay Mesh Generation. Technical report, 1999.

[95] J.R. Shewchuk. Mesh generation for domains with small angles. In *Sixteenth Annual Symposium on Computational Geometry*, pages 1–10, Hong Kong, 2000.

[96] J.R. Shewchuk. Sweep Algorithms for Constructing Higher-Dimensional Constrained Delaunay Triangulations. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pages 350–359, Hong Kong, June 2000.

[97] J.R. Shewchuk. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. In *Proceedings of 11th International Meshing Roundtable*, pages 193–204, September 2002.

[98] JR Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry-Theory and Applications*, 22(1-3):21–74, MAY 2002.

[99] J.R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *International Meshing Roundtable*, 2002.

[100] J.R. Shewchuk. General-dimensional constrained Delaunay and constrained regular triangulations, I: Combinatorial properties. *Discrete & Computational Geometry*, 39(1-3):580–637, Mar 2008.

[101] K. Shimada and D.C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 409–419, Salt Lake City, 1995.

[102] K Shimada, N Mori, T Kondo, T Itoh, K Kase, and A Makinouchi. Automated mesh generation for finite element analysis of sheet metal forming. *International Journal of Vehicle Design*, 21(2-3):278–291, 1999.

[103] K. Shimada, A. Yamada, and T. Itoh. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In *6th International Meshing Roundtable*, pages 375–390, 1997.

[104] H. Si. Tetgen - a quality tetrahedral mesh generator. http://tetgen.berlios.de/.

[105] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3), 1978.

[106] R. Sibson. *A Brief Description of Natural Neighbor Interpolation*. John Wiley, 1981.

[107] J.M. Sullivan and Y. Zhang. Integrated Surface and Ground Water Modeling for Contaminant Fate and Transport Predictions. *J. of Hydro. Sci. and Tech*, 17(1-4):341–350, 2001.

[108] M. Tanemura, T. Ogawa, and N. Ogita. A new algorithm for three-dimensional Voronoi tessellation. *Journal of Computational Physics*, 51:191–207, 1983.

[109] A Ungor. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. In *LATIN 2004: Theoretical Informatics*, volume 2976, pages 152–161. Springer - Verlag Berlin, 2004.

[110] Z. J. Wang. A quadtree-based adaptive cartesian/quad grid flow solver for navier-stokes equations. *Computers & Fluids*, 27(4):529–549, MAY 1998.

[111] D.F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *Computer Journal*, 24(2):167 – 172, 1981.

[112] N.P. Weatherill. The Integrity of Geometrical Boundaries in the 2-Dimensional Delaunay triangulation. *Communications in Applied Numerical Methods*, 6(2):101–109, Feb 1990.

[113] N.P. Weatherill. Delaunay Triangulation in Computational Fluid Dynamics. *Computers and Mathematics with Applications*, 24(5/6):129–150, September 1992.

[114] N.P. Weatherill and O. Hassan. Efficient 3-Dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37(12):2005–2039, Jun 1994.

[115] Ziji Wu. *Accurate and Efficient Three-Dimensional Mesh Generation for Biomedical Engineering Applications*. PhD thesis, Worcester Polytechnic Institute, April 2001.

[116] Ziji Wu and J.M. Sullivan. Multiple material marching cubes algorithm. *Int. J. Numer. Methods Eng.*, 58(2):189 – 207, 14 Sept. 2003.

[117] P.K. Yalavarthy, H. Dehghani, B.W. Pogue, and K.D. Paulsen. Critical computational aspects of near infrared circular tomographic imaging: Analysis of measurement number, mesh resolution and reconstruction basis. *Optics Express*, 14(13):6113–6127, Jun 2006.

[118] D. Yan, A. Khajepour, and R. Mansour. Modeling of two-hot-arm horizontal thermal actuator. *Journal of Micromechanics and Microengineering*, 13(2):312–322, Mar 2003.

[119] Mark A. Yerry and Mark S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, 20(11):1965 – 1990, 1984.

[120] P. D. Zavattieri, E. A. Dari, and G. C. Buscaglia. Optimization strategies in unstructured mesh generation. *International Journal for Numerical Methods in Engineering*, 39(12):2055–2071, Jun 1996.

[121] J. Q. Zhang, J. M. Sullivan, and H. R. Ghadyani. MRI guided 3D mesh generation and registration for biological modeling. *Journal of Computing and Information Science in Engineering*, 5(4):283–290, 2005.