

**A COMBINED QUADTREE/DELAUNAY METHOD  
FOR 2D MESH GENERATION**

A Thesis Presented

by

SIMON TANG

Submitted to the Graduate School of the  
University of Massachusetts Amherst in partial fulfillment  
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2012

Electrical and Computer Engineering

© Copyright by Simon Tang 2012

All Rights Reserved

# A COMBINED QUADTREE/DELAUNAY METHOD FOR 2D MESH GENERATION

A Thesis Presented

by

SIMON TANG

Approved as to style and content by:

---

Marinos N. Vouvakis, Chair

---

Eric Polizzi, Member

---

Weibo Gong, Member

---

Christopher V. Hollot, Department Head  
Electrical and Computer Engineering

## ABSTRACT

# A COMBINED QUADTREE/DELAUNAY METHOD FOR 2D MESH GENERATION

MAY 2012

SIMON TANG

B.Sc., UNIVERSITY OF MASSACHUSETTS AT AMHERST

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Marinos N. Vouvakis

Unstructured simplicial mesh is an integral and critical part of many computational electromagnetics methods (CEM) such as the finite element method (FEM) and the boundary element method (BEM). Mesh quality and robustness have direct impact on the success of these CEM methods.

A combined quadtree/Delaunay 2D mesh generator, based on the early work of Schroeder (1991, PhD), is presented. The method produces a triangulation that approximates the original geometric model but is also topologically consistent. The key advantages of the method are: (a) its robustness, (b) ability to create a-priori graded meshes, and (c) its guaranteed mesh quality.

The method starts by recursively refining the grid and using a 2:1 balanced quadtree data structure to index each cell. Once the quadtree grid is refined at a user-defined level associated with each geometrical model topological entity, each cell in the grid is successively triangulated using the Delaunay method. Finally, the

method handles some modeling errors by merging vertices and allowing overlapped faces.

# TABLE OF CONTENTS

	Page
<b>ABSTRACT</b> .....	<b>iv</b>
<b>LIST OF FIGURES</b> .....	<b>viii</b>
 <b>CHAPTER</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Problem Statement .....	1
1.2 Significance .....	4
1.3 Challenges .....	5
1.4 Literature Review .....	6
1.4.1 Delaunay Triangulation .....	6
1.4.2 Advancing Front .....	12
1.4.3 Grid and Quadtree Mesh Generation .....	14
1.5 Contributions .....	20
<b>2. APPROACH</b> .....	<b>23</b>
2.1 Algorithm .....	23
2.1.1 Spatial Partitioning via Tree Data Structure .....	24
2.1.1.1 Inserting Vertices .....	24
2.1.1.2 Mesh Control Parameter .....	30
2.1.1.3 2:1 Balance Condition .....	31
2.1.1.4 Inserting Edges .....	31
2.1.1.5 Inserting Faces .....	33
2.1.2 Successive Triangulation .....	34
2.1.2.1 Conforming Delaunay Triangulation .....	36
2.1.2.2 Delaunay Refinement .....	37

2.1.3	Mesh Quality Improvement . . . . .	42
2.1.4	Quality Guarantees . . . . .	44
2.2	Implementation . . . . .	46
2.2.1	Data Structures . . . . .	46
2.2.2	Functions . . . . .	49
<b>3.</b>	<b>RESULTS . . . . .</b>	<b>51</b>
3.1	Delaunay Triangulation . . . . .	54
3.2	Uniform Grid . . . . .	56
3.3	Non-uniform Refinement . . . . .	59
3.4	Merge Operations . . . . .	59
3.5	Realistic Two-Dimensional Models . . . . .	66
3.6	Discretized Curve and Smooth Curves . . . . .	70
3.7	Handling Models from a Poor Quality CAD Tool . . . . .	74
3.8	Elements vs. CPU Time . . . . .	76
3.9	Delaunay Refinement as Successive Triangulation Method . . . . .	78
<b>4.</b>	<b>CONCLUSION AND FUTURE WORK . . . . .</b>	<b>82</b>
4.1	Conclusion . . . . .	82
4.2	Future Work . . . . .	83
	<b>BIBLIOGRAPHY . . . . .</b>	<b>85</b>

## LIST OF FIGURES

Figure	Page
1.1 Relationship between the geometry, attributes, and the topology.....	2
1.2 The two shapes are topologically consistent although the geometry may be different. ....	3
1.3 Triangulation is topologically consistent with the domain. ....	3
1.4 Stages of mesh-based scientific computing method. The highlighted stage is the focus of this research. ....	4
1.5 Modeling error in two touching squares. However, the solid modeling tool made two overlapping squares, instead of the touching squares. ....	5
1.6 Missing Edges in Delaunay Triangulation [13]. ....	7
1.7 Bowyer-Watson algorithm [13]. ....	8
1.8 Degeneracy in Bowyer-Watson algorithm [13]. ....	8
1.9 Lawson's algorithm [13]. ....	9
1.10 Constrained and conforming triangulation [16]. ....	10
1.11 Vertex $a$ encroaches segment $s_1$ [15]. ....	11
1.12 Ruppert's refinement of skinny triangles [15]. ....	12
1.13 The front in this iteration is colored in red. ....	13
1.14 Advancing front walkthrough [9]. ....	13
1.15 Quadtree for referencing subdivision of a 2D space. The graph representations of the Figures (a),(b), and (e) are shown below their corresponding figures. ....	15



1.16	Crowded quadrants. Each example violates one of the three conditions. The crowded are colored in red. (a) Two vertices in a quadrant. (b) Vertices are within $2\sqrt{2}l$ where $l$ is the [larger] length of the quadrant they are in. (c) An extended neighbor is subdivided. ....	17
1.17	Warping of the grid to vertices of the domain. ....	18
1.18	Warping of grid to the edges of the domain [9]. ....	19
1.19	A domain with jagged interfaces and a domain with flat interfaces. ....	21
2.1	Overview of the proposed combined quadtree/Delaunay triangulation. (a) Space partitioning of the geometric model, and classification of quadrants (interior/boundary/exterior). (b) Independent constrained DT triangulation of each quadrant. (c) Elimination of poor quality triangles by merging select quadtree/geometry intersection point into the quadtree vertices. ....	23
2.2	Geometric model with attributes and the respective root quadrant in the spatial partitioning. The vertex mesh control parameter (vMCP) is a mesh control parameter assigned to a vertex by the user. ....	24
2.3	Insertion of geometric vertices, quadrant subdivision, computation of the corresponding intersections, and 2:1 balance condition. ....	28
2.4	Mesh control parameter (MCP) on different topological entities. (a) Vertex. (b) Edge. (c) Face. ....	31
2.5	Benefits from enforcing the 2:1 balancing condition on the quadtree spatial partitioning. Additional vertices that are added due to the 2:1 balance condition are colored in red. ....	32
2.6	Insertion of an edge (curve) with eMCP=3 into the quadtree space partitioning. Note that while the quadrants are refined, the new quadrants must be re-classified as boundary (yellow). ....	33
2.7	Insertion of a face (circular disc) with fMCP=4 into the quadtree space partitioning. Note that while the quadrants are refined, they must be classified as interior, exterior and boundary quadrants. ....	34
2.8	Overview of the successive triangulation steps through an example ....	35

2.9	The right quadrant introduces the red point. The triangulation on left does not match the triangulation on the right on the interface, or the green edge. ....	36
2.10	Triangulated template of a quadrant .....	37
2.11	Preservation of geometric edges in a quadrant by stitching. The missing blue edge on the left figure is recovered on the right by recursively inserting the intersection vertex of the edge with the existing mesh edges via the Bowyer-Watson algorithm. ....	37
2.12	Overview of the successive triangulation method with a Delaunay refinement algorithm. (a) Space partitioning of the geometric model. (b) Refinement of the quadrant boundaries. (c) Delaunay refinement of each quadrant. ....	38
2.13	Possible non-conforming meshing between quadrants if the Delaunay refinement is independently applied on each neighboring quadrant.....	39
2.14	Triangulation of an empty quadrant with the edges on the perimeter of the quadrant as constrained edges.....	39
2.15	Refinement of the quadrant boundaries. Vertices on the quadrant boundaries are no more than the threshold length $l_t$ apart as defined in (2.2). ....	40
2.16	Combined quadtree and successive triangulation, where the triangulation is performed by the Delaunay refinement from the Triangle [16] code. Delaunay refinement algorithm as a successive triangulation method. The edges along the perimeter of the quadrant are set to be constrained edges and subdivided until it is smaller than the threshold length. ....	41
2.17	Elimination of poor quality (skinny) triangles by merging select vertex points from resulting from the intersection of the quadrant and geometry to a quadrant vertex. ....	42
2.18	Merge cases that is considered in this research. The top figures show the states before the merge process, and the bottom figures show the states after the merge process. The blue and green vertices are within the merging distance. The blue vertices are merged to the green vertices. ....	43

2.19	Case I, where the present merge strategy fails to filter poor quality triangles. $\lim_{w \rightarrow 0} Q(\triangle) = 0$ , where $\triangle$ is the red triangle and $Q$ measures the quality of the triangle. ....	44
2.20	Case II, where the present merge strategy fails to filter poor quality triangles. $\lim_{w \rightarrow 0} Q(\triangle) = 0$ , where $\triangle$ is the red triangle and $Q$ measures the quality of the triangle. The yellow areas marks the maximum merging distance of their corresponding centers. ....	45
2.21	Merge candidate for triangles based on Condition 2.1.1. The red triangles will be removed to provide a low bound on the mesh quality. ....	47
3.1	The compilation above composes all of the possible graded, interior, and exterior quadrants in a triangulation. These quadrants are very common in this work. Thus, the frequency of the common triangles is high because they can be found in these quadrants. ....	52
3.2	The list above calculates the quality $Q$ of each common triangle. ....	53
3.3	The conforming Delaunay triangulation of a square is shown on the left, and its quality distribution is shown on the right. ....	54
3.4	Conforming Delaunay triangulation of PUMA. ....	55
3.5	Smooth circular disk that is used to create the triangulations shown in Figures 3.6 and 3.7. ....	57
3.6	Uniform quadtree triangulation of a circular disk inside a box as depicted in Figure 3.5. (a) Triangulation with uniform (balanced) quadtree spatial partitioning of level 3 (b) Mesh quality distribution. ....	57
3.7	Uniform quadtree triangulation of a circular disk inside a box as depicted in Figure 3.5. (a) Triangulation with uniform (balanced) quadtree spatial partitioning of level 5 (b) Mesh quality distribution. ....	58
3.8	Non-uniform refinement. (a) Node refinement, where MCP is assigned to all four nodes. (b) Edge refinement, where MCP is assigned at the four edges of the square. and (c) Face refinement, where the MCP is assigned on the face of the square (d)Histogram of the resulting mesh qualities. ....	60

3.9	Poor quality triangles (highlighted in the yellow area) from the uniform triangulation of a circle in Figure 3.7 in a transparent yellow area. . . . .	61
3.10	Improving the mesh quality of Figure 3.9 by merging geometry points to the quadtree corners and effectively eliminating the skinny triangles, but introducing some geometric error. . . . .	61
3.11	The same circle as in Figure 3.7 is uniformly triangulated with the same mesh control parameter, 5. In this case, the point merges are enabled and its merge tolerance is 0.10. The triangulation is on the left, and its quality distribution is on the right. . . . .	62
3.12	Plot of the relationship between approximation error and merge tolerance for the circle model. . . . .	63
3.13	Effect of merge on mesh quality. (a) There are no calculated quality cut-off when the merge tolerance is 0. (b) The quality cut-off is at 0.1694. (c) The quality cut-off is at 0.3149. (d) The quality cut-off is at 0.4133. . . . .	65
3.14	Triangulation of a torus. The mesh generator can mesh shapes with holes and remove them from the triangulation. (a) Geometric model of a torus with a $fMCP = 5$ . (b) Triangulation of a torus. . . . .	66
3.15	Triangulation of traces on printed circuit board (PCB). The mounting holes are removed from the triangulation. . . . .	67
3.16	Different mesh control parameters on different topological elements. (a) Model of balanced antipodal Vivaldi antenna (BAVA) [35] (b) Triangulation of BAVA. . . . .	67
3.17	Model of the flare piece from a Vivaldi antenna. . . . .	68
3.18	Topological inconsistencies resulting from poor usage of MCP and merge tolerance. (a) Case where the $merge=0.2$ and the $eMCP=6$ , leading to a topologically consistent mesh. (b) Case where the $merge=0.2$ and the $eMCP=4$ , leading to a topologically inconsistent mesh. . . . .	69

3.18	Meshing of discretized vs. smooth parametric curves. In all cases, the merge tolerance is 0.3 and the edge MCP=6. (a) Mesh of discrete parametric curve. (b) Zoomed view of a region around the curve (a). (c) Mesh of the smooth parametric curve (d) Zoomed view of a region around the curve in (c). (e) Side-by-side comparison of the mesh quality histograms, showing the smooth parametric curve model has the better mesh quality.....	72
3.19	Model of the Vivaldi antenna, with both conducting (red) and dielectric (green) pieces. ....	73
3.20	User enabled merge option for a Vivaldi antenna. (a) Merging all edges of the model to the quadtree nodes. (b) selectively merging only the exponential flare and circular region edges. ....	74
3.21	Model created from a poor quality CAD. (a) Two touching squares are created in a poor quality CAD tool. The circle zooms in on the bottom region of the “supposedly” touching edges. (b) Conforming Delaunay triangulation of the model shown in Figure (a). (c) Delaunay refinement triangulation of the model shown in Figure (a). (d) The two squares are touching in the triangulation because the vertices along the touching edges merges to a vertex on the other square. The two squares have $fMCP = 4$ . (e) The two squares are touching in the triangulation because the vertices along the touching edges merges to a vertex on the other square. The two squares have no mesh control parameters. ....	75
3.22	Model and triangulation of the PUMA. ....	77
3.23	CPU time vs. number of generated elements for the PUMA example. The figure also compares the cases with and without merge. Both curves for the spatial partitioning and triangulation are $O(n \log n)$ . ....	77
3.24	Mesh of a Vivaldi flare using a Delaunay refinement algorithm as the successive triangulation procedure. ....	80
3.25	Graded mesh of a Vivaldi flare using a Delaunay refinement algorithm as the successive triangulation procedure. ....	81

# CHAPTER 1

## INTRODUCTION

### 1.1 Problem Statement

The goal of this thesis is to automatically produce a topologically compatible two-dimensional mesh approximation (triangulation) given a geometric model and user defined attributes such as boundary conditions, a merge tolerance, and a mesh control parameter on each topological element. Topological entities in a geometrical representation are the faces, edges, and vertices from the geometric representation.

To fully understand the role of mesh generation as stated above, it is important to outline the nature of a geometric model and the necessary topological consistency between the geometric model and the mesh approximation.

A geometric model  $D = \{G, T, A\}$  is a representation that describes the geometry  $G$ , topology  $T$ , and the associated attributes  $A$ , of all objects/shapes in the model. The geometry  $G$  is a mathematical description of position in Euclidean. The most common description are nodes of straight line segments, Bézier curves or non-uniform rational B-splines (NURBS). Topology is the connectivity information between geometric entities, and it is invariant under geometrical transformations. The topology of an object is represented as a set of hierarchically ordered elements  $T = \{V, E, F\}$ , where  $V$  is a vertex,  $E$  is an edge and  $F$  is a face. Finally, attributes are information associated with each topological entity, and they can be related to the mesh, such as the mesh refinement size, color, and boundary conditions for partial differential equation (PDE) solvers.

The most important property of a “valid” mesh in the context of PDE solvers such as finite element method (FEM) is the topological consistency between the mesh  $M$  and the original geometric model  $D$ , meaning that each topological entity of the mesh should be associated (linked) to one topological entity of the geometric model, and that these mesh entities should form a proper topological hierarchy. Figure 1.1 shows this hierarchy.

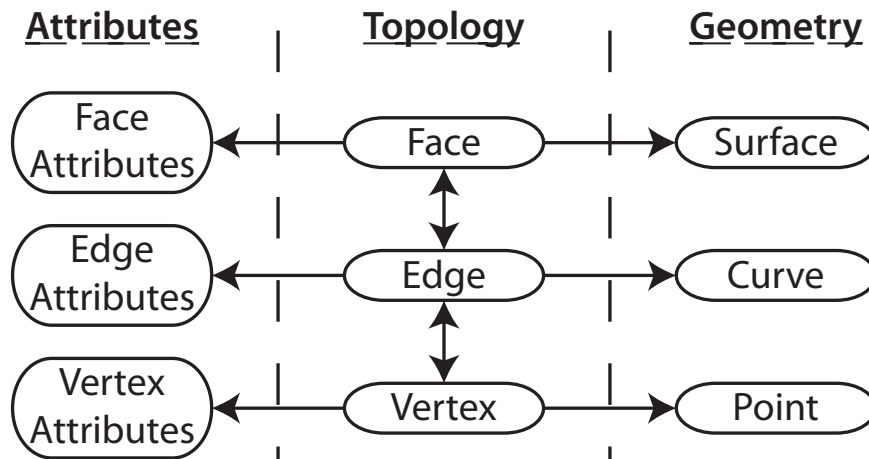


Figure 1.1: Relationship between the geometry, attributes, and the topology.

Figure 1.2 shows an example of two shapes that are topologically consistent. The right has the same topology as the left of matching color. Each topological element on the right has a different geometry than its topological element on the left. However, they are topologically consistent because they share the same neighbors. For example, the green edge on right figure is adjacent to a blue edge and a yellow edge, which matches the adjacency of the green edge on the left figure. Finally, Figure 1.3 shows an example where the mesh is topological consistent with the geometric model.

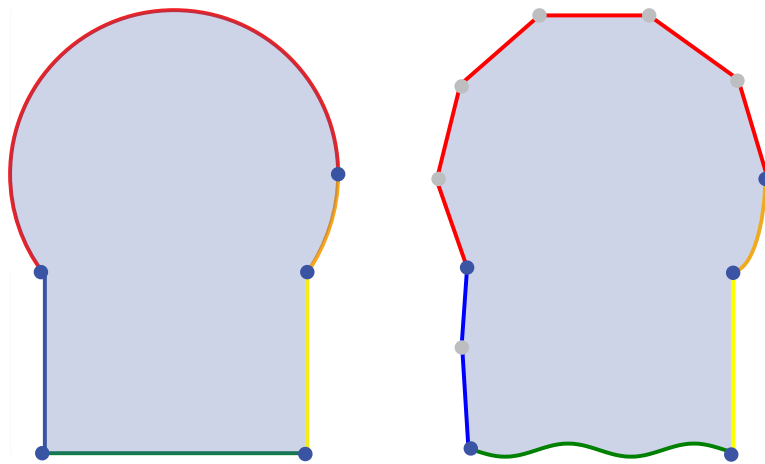
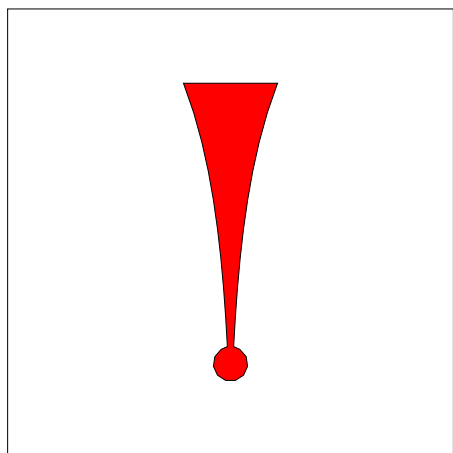
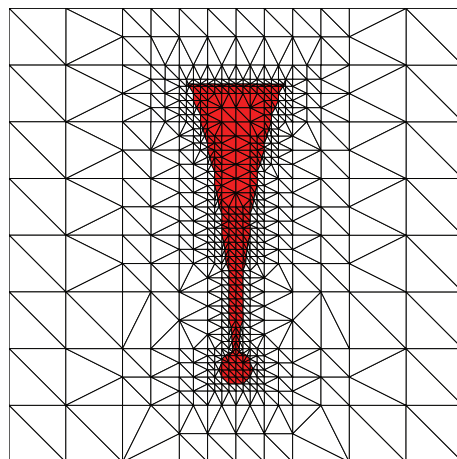


Figure 1.2: The two shapes are topologically consistent although the geometry may be different.



(a) Geometric domain.



(b) Mesh approximation.

Figure 1.3: Triangulation is topologically consistent with the domain.



## 1.2 Significance

In the modern engineering design processes, engineers model their designs on computer aided design (CAD) software and run simulations of their designs. These engineering simulations are subject to the law of physics. Using computational methods to solve the equations that are govern by the law of physics on a given problem is automatic, cheap, and fast in comparison to approximation and analytical methods. Finite element method (FEM) [1, 2, 3] and boundary element method [4, 5, 6, 7] are two computational methods that requires a mesh. Figure 1.4 shows an overview of the process in mesh-based scientific computing methods. The focus of this research is to generate a high quality mesh for these methods.

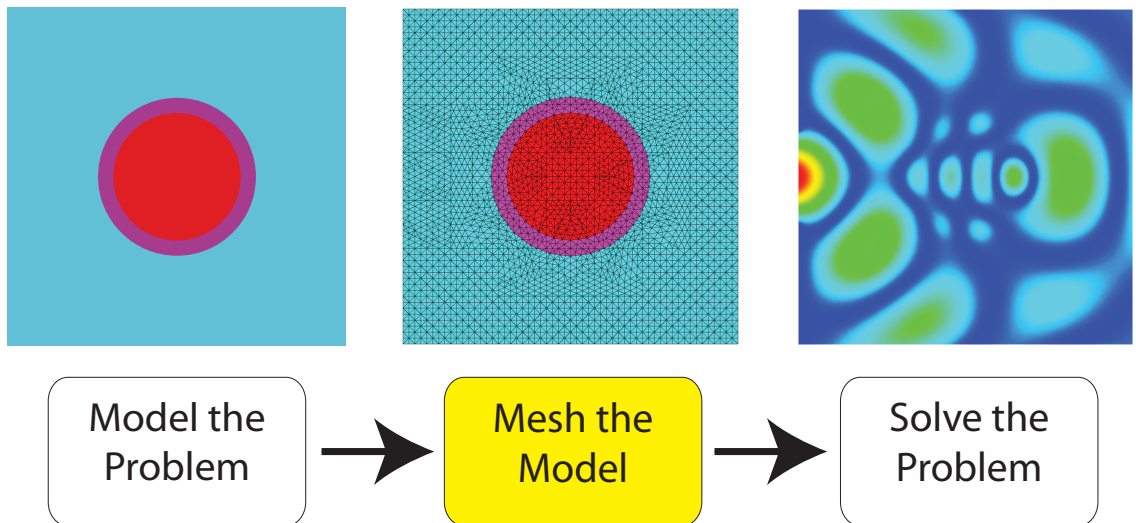


Figure 1.4: Stages of mesh-based scientific computing method. The highlighted stage is the focus of this research.

In all mesh-based scientific computing methods, the shape of the elements in the mesh directly affect the numerical efficiency and accuracy of the method. For example, skinny triangles are undesirable because they lead to ill-conditioned matrices and have poor interpolation error estimates [8]. Moreover, the computational resources used by all mesh-based scientific computing methods are proportional to the size of the mesh. Thus, greater number of elements will require more run time and memory. Therefore,

it is important to avoid having many elements in an area where many elements are not needed.

### 1.3 Challenges

A common problem of mesh algorithms is the robustness of handling models that are created from poor quality CAD tools. This is commonly due to problems with tolerances, such as the one shown in Figure 1.5. The two squares in Figure 1.5 are supposed touching. However, due to problems with the tolerances, there is a spacing between the bottom-right vertex on the red square and the bottom-left vertex on the green square. A mesh generator may treat these two vertices as two topologically different vertices. As a result, these mesh generator may crash, produce poor quality elements, and produce too many elements. As a result, FEM users spend a lot of time fixing their models to get their mesh software to correctly produce a mesh of the model.

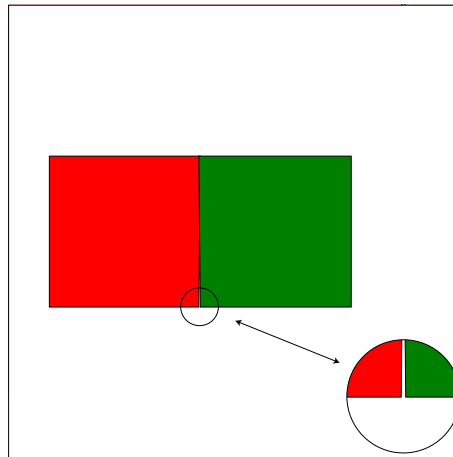


Figure 1.5: Modeling error in two touching squares. However, the solid modeling tool made two overlapping squares, instead of the touching squares.

Mesh size is another common challenge in mesh generation. Generally, with greater number of elements, the quality of the elements will improve. The most well-known mesh generation algorithms in the literature has mathematical quality

guarantees. Some of these algorithms are proven to be size-optimal, which means the number of elements in the triangulation is a constant from having the minimum number of elements for a given quality guarantee. Often, these size-optimal meshes create too many elements in practice. Though, they may be size-optimal, but the constant from having the minimum elements for a quality guarantee is very large.

## 1.4 Literature Review

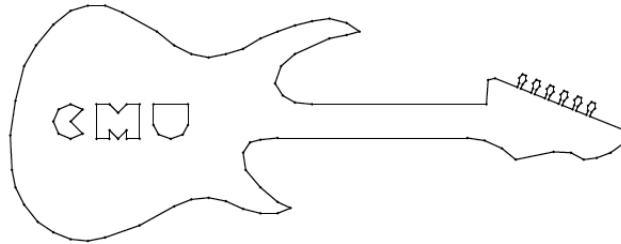
Over the years, many researches have tried to address the challenges of automatic mesh generation with simplexes in both two and three dimensions. Although this thesis deals with 2D meshes, the literature review will include important works from 3D meshes, too. There are three common categories of automatic mesh generations: Delaunay triangulation, advancing front, and grid/quadtrees mesh generation. They are outlined in the sections below.

### 1.4.1 Delaunay Triangulation

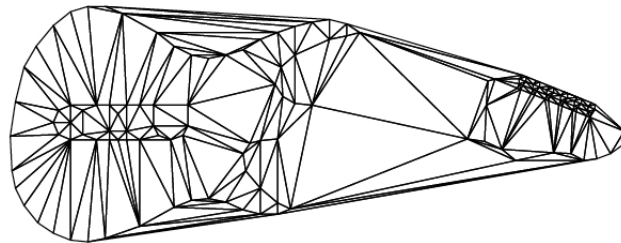
A Delaunay triangulation is a triangulation that meets the Delaunay condition. Under the Delaunay condition, the circumscribed circle of each triangle in the two-dimensional mesh does not contain any points on its interior. The Delaunay triangulation has a special property that the minimum angle is maximized. As mentioned in the previous section, small angles can produce numerical inaccuracies for scientific computing applications. Thus, this is the motivation for the use of a Delaunay triangulation algorithm. However, this property does not extend to the three-dimensional case [9].

Two of the most commonly known Delaunay triangulation algorithms are Lawson's algorithm [10] and Bowyer-Watson algorithm [11, 12]. Delaunay triangulation algorithms read in a set of vertices and produce a Delaunay triangulation from these vertices. Delaunay triangulation algorithms do not read in any edges or faces. There-

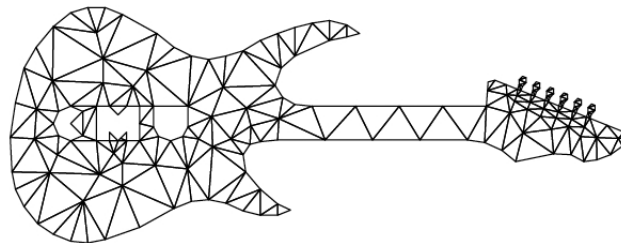
fore, edges and faces from the geometric domain may not be preserved, or is not topologically represented by a set of mesh elements, in the triangulation. Figure 1.6(b) illustrates a Delaunay triangulation of the guitar and its missing edges in the triangulation. In Figure 1.6(c), the edges are preserved.



(a) Geometric model



(b) Delaunay triangulation



(c) Edges are preserved

Figure 1.6: Missing Edges in Delaunay Triangulation [13].

Both algorithms are incremental algorithms. At each increment, the Delaunay triangulation inserts a point from the set of vertices given at input and is updated to include an inserted point. The two algorithms terminate when all points are inserted.

At each increment, Bowyer-Watson algorithm finds all of the triangles whose circumscribed circles contain the inserted point. All of these triangles will be deleted and form a convex polygon. The convex polygon will be triangulated where all trian-

gles incident on the inserted point. The Bowyer-Watson algorithm is demonstrated in Figure 1.7.

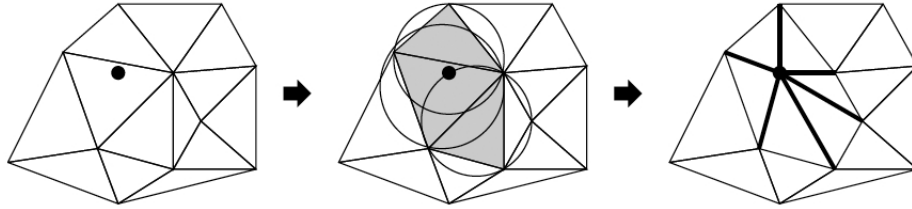


Figure 1.7: Bowyer-Watson algorithm [13].

The Bowyer-Watson algorithm is less robust in the case of degeneracies, where the vertices are exactly on the perimeter of the circumscribed circle. Figure 1.8 depicts an example of this degeneracy. However, the algorithm can be easily generalized to higher dimensions. The three dimensional Bowyer-Watson algorithm substitutes a circumscribed sphere for a circumscribed circle in the two-dimensional Bowyer-Watson algorithm.

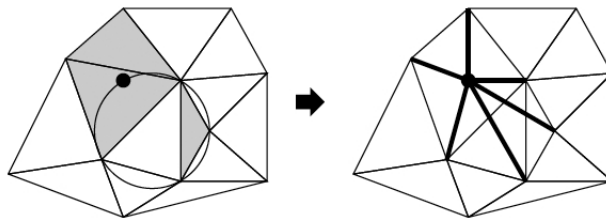


Figure 1.8: Degeneracy in Bowyer-Watson algorithm [13].

In summary, the Bowyer-Watson algorithm is a very intuitive approach and can be generalized to arbitrary dimensionality, but it is not very robust because of its issues with degeneracies. The time complexity of the Bowyer-Watson algorithm is  $O(n \log n)$ .

On the other hand, Lawson's algorithm first locates the triangle that contains the inserted point and divides the triangle into three at each increment. Then, the algorithm flips edges until the current triangulation is Delaunay by checking Delaunay

condition on every edge flip. The check is performed on the two triangles that had been edge flipped. If the triangle does not satisfy the Delaunay condition, then the triangle and the adjacent triangle that are incident on the encroached vertex will be edge flipped. The walkthrough of Lawson’s algorithm on an example is depicted on Figure 1.9.

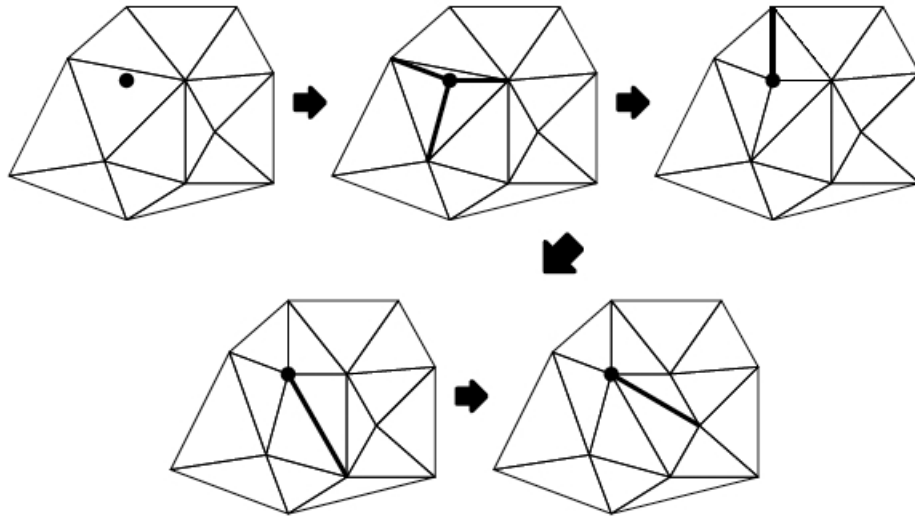


Figure 1.9: Lawson’s algorithm [13].

Lawson’s algorithm can be very advantageous because of its robustness. Unlike Bowyer-Watson algorithm, degeneracies do not have any negative impacts on Lawson’s algorithm. In comparison to Bowyer-Watson algorithm, Lawson’s algorithm is theoretically harder to generalize to arbitrary dimensionality. Like Bowyer-Watson algorithm, the time complexity of Lawson’s algorithm is  $O(n \log n)$ .

There are two types of Delaunay triangulation algorithms that preserve the edges from the domain: constrained Delaunay triangulation[14] and conforming Delaunay triangulation[14]. The conforming Delaunay triangulation introduces points that are not originally from the input point set for the purpose of preserving the edges. Points that were not part of the input point set are called Steiner points[15]. The constrained Delaunay triangulation does not introduce any Steiner points to the triangulation.

It is not always possible to achieve a Delaunay triangulation with constrained edges and without inserting Steiner points. Therefore, the constrained Delaunay triangulation may not always be a Delaunay triangulation. Examples of a constrained and a conforming Delaunay triangulation are shown in Figure 1.10.

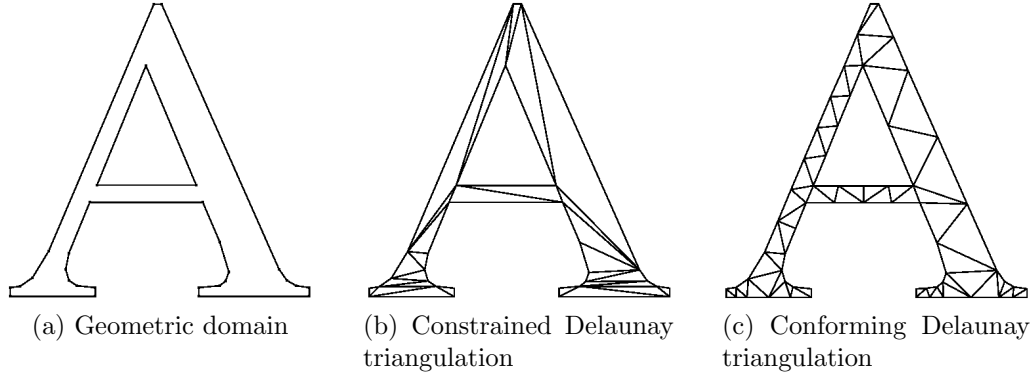


Figure 1.10: Constrained and conforming triangulation [16].

Although conforming Delaunay triangulation itself triangulates the domain, preserves its topological elements, and maximizes the minimum angle in the triangulation, skinny triangles may still result in the triangulation. Delaunay refinement algorithms [15, 17, 18] strategically insert Steiner points into the triangulation to improve the quality. These Steiner points will cause these poor quality elements to be deleted and construct new elements to fill the empty spaces that are resulted from deleting elements. One of the commonly known refinement algorithm is Ruppert's Delaunay refinement algorithm [15]. His paper presents a two dimensional Delaunay refinement algorithm and proves that it have some mathematical guarantees. His algorithm can be generalized to a three dimensional refinement algorithm. However, the quality guarantee may not extend to its three dimensional generalization.

Ruppert's Delaunay refinement algorithm reads in a geometric domain and a lower bound on the angles. The algorithm produces a conforming Delaunay triangulation with a minimum angle that is specified at input if it terminates. The algorithm stitches missing edge segments by inserting Steiner points at the midpoint of the

missing segments. Similarly, any encroached edges in the triangulation will be split the same way as the missing segments. An encroached segment is a segment from the triangulation whose diametral circle that has a point from the triangulation in its interior or a missing segment. A diametral circle of a segment is a circle whose center is the midpoint of the segment and its radius is half of the length of the segment. Figure 1.12 demonstrates an encroached segment and a diametral circle.

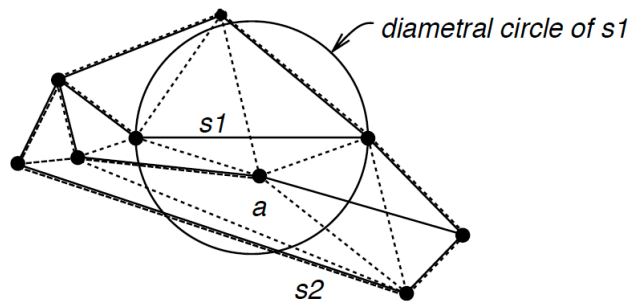


Figure 1.11: Vertex  $a$  encroaches segment  $s_1$  [15].

Triangles with angles smaller than the lower bound are refined by inserting the Steiner point at its circumscribed center. Such triangles are considered as skinny triangles in this algorithm.

Ruppert's Delaunay refinement algorithm offers guarantees on quality and size-optimality, which means the number of elements in the triangulation are within a constant factor from the minimum number of elements to meet the same bound. These guarantees are mathematically proven[15]. It is also noted in Ruppert's original paper that the algorithm fails to terminate for lower bounds greater than  $30^\circ$ . His paper mathematically proves that the algorithm terminates for any lower bounds less than  $20^\circ$ .

In conclusion, Delaunay triangulations are great for two dimensional problems because it maximizes the minimum angle. However, this property does not extend to three dimensional problems. Delaunay refinement algorithms are generally fast and



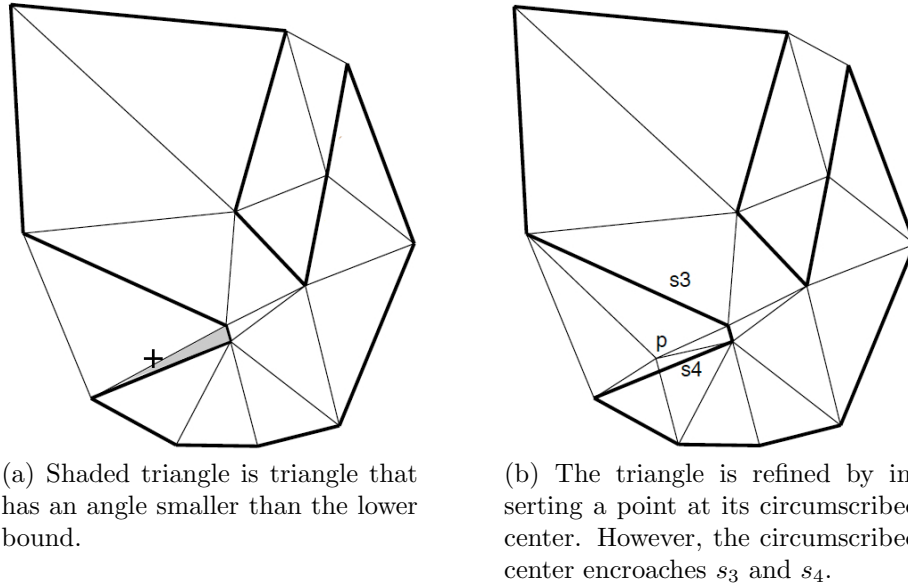


Figure 1.12: Ruppert's refinement of skinny triangles [15].

efficient. Finally, it generate large meshes, especially if there are small angles in the geometric model.

### 1.4.2 Advancing Front

Advancing front algorithms [19, 20, 21, 22] start by discretizing the boundaries of the domain. These set of edges are called the front. These algorithms insert Steiner points at places that would construct the best quality triangles possible based on the current front. As new triangles are created, edges are obscured and removed from the front. Figure 1.13 shows an example of a front in an advancing front algorithm. In this figure, the front is colored in red, and the triangles are created based on the previous front, which is the perimeter of the domain. Additionally, new edges are created from creating triangles. Therefore, the front will include the new edges. Based on the new front, the algorithm constructs its next set of triangles. The algorithm pushes the front toward the interior of the domain until the whole domain is triangulated. Figure 1.14 demonstrates the advancing front process.

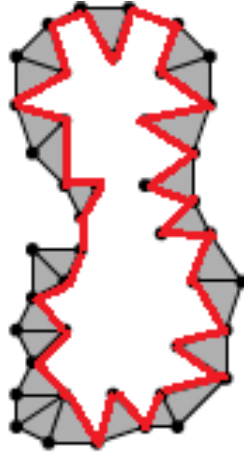


Figure 1.13: The front in this iteration is colored in red.

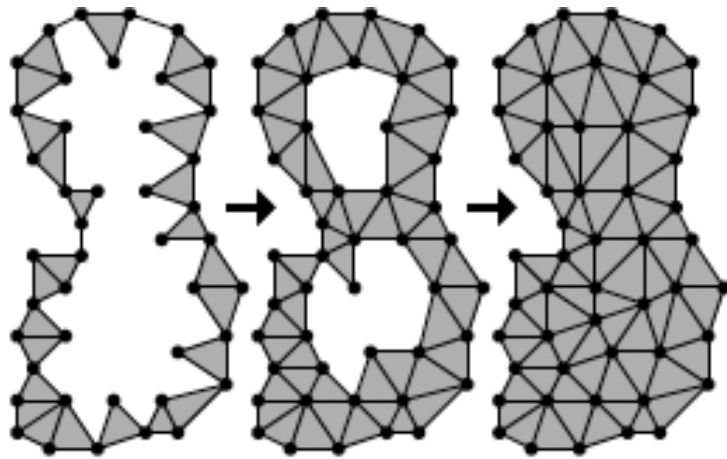


Figure 1.14: Advancing front walkthrough [9].

Before the constructing the last set of triangles, the algorithm may attempt to insert vertices in the interior of another constructed triangle because this will result in the best quality triangle based on the front. Constructing triangles with vertices like these will cause triangles to overlap. This occurrence is called a collision. When there is a collision, some advancing front algorithm will use an existing vertices to construct the new triangle to fill the space, and others may insert it and use a Delaunay triangulation method or other some methods to resolve the collision [20].

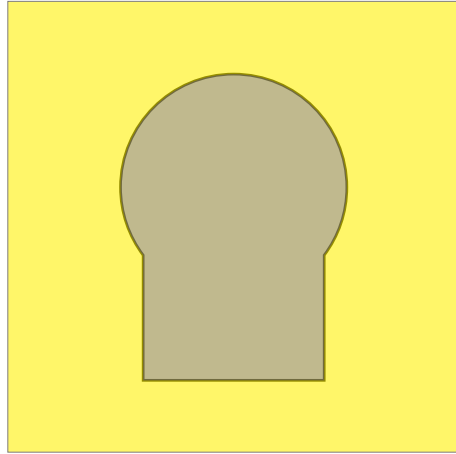
The poorest quality elements can be found in the interior where the fronts collide, and the best are found along the perimeter on the domain. In general, the advancing front algorithms create very smooth meshes. Of the three categories, advancing front is the least robust.

### 1.4.3 Grid and Quadtree Mesh Generation

Grid and quadtree mesh generation [23, 24, 25, 26, 27, 28, 29, 30] subdivides the problem into smaller subproblems before the triangulation process, and it uses a tree data structure to reference each of the subdivided spaces. A quadtree data structure, which has a maximum of four children, is used to reference the subdivided spaces for two dimensional problems. Figure 1.15 shows the referencing from quadtree to the subdivided spaces.

Grid-based mesh generation algorithms place all geometric models into a root quadrant, or a bounding box. Using Figure 1.15(a) as an example, the keyhole shape is the geometric model in this problem, and the yellow square that encloses the keyhole model is the root quadrant. The size of the root quadrant may vary from one grid-based mesh generation algorithm to another. For this example, this root quadrant is graphically represented as the yellow node in Figure 1.15(c), and the single yellow node is the quadtree configuration of Figure 1.15(a) because there are no subdivisions. When the yellow quadrant in Figure 1.15(a) is subdivided, it creates four quadrant as shown in Figure 1.15(b), which are colored in green, navy-blue, orange, and red. Each of the quadrant is a represented as a node (of matching color) in the graph representation. These four nodes branch off from the yellow node because the four corresponding quadrants derived from subdividing the yellow quadrant.

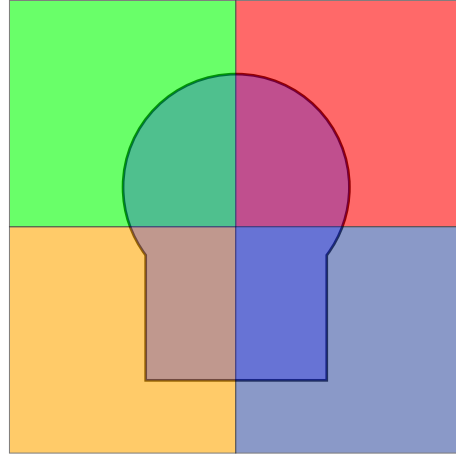
“Provably Good Mesh Generation” [23], by Bern, Eppstein, and Gilbert, is one of the commonly known quadtree approach. This is a two-dimensional mesh algorithm, and it can be generalized for three dimensions. This specific algorithm offers math-



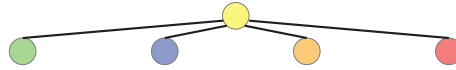
(a) Quadrant that encloses the model is created.



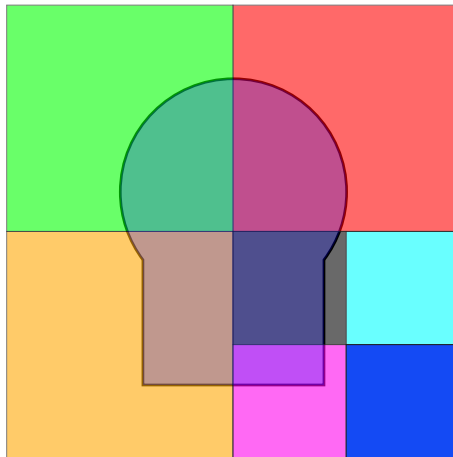
(c) The yellow node is a graph representation of the yellow quadrant.



(b) The root quadrant is subdivided into four smaller quadrants.



(d) In the graph representation, four children nodes, each correspond to a child quadrant of matching color, branch off from the node that represents the quadrant that is subdivided, which is the yellow node.



(e) Navy blue quadrant is subdivided.



(f) Four children nodes, each correspond to a child quadrant of matching color, branch off from the navy-blue node, which is the graph representation of the quadrant that is subdivided.

Figure 1.15: Quadtree for referencing subdivision of a 2D space. The graph representations of the Figures (a),(b), and (e) are shown below their corresponding figures.

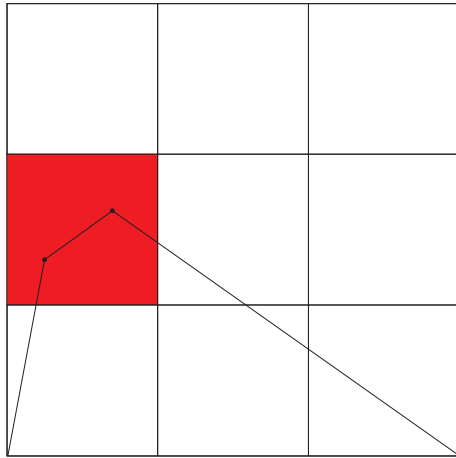
emathical bounds on quality and size-optimal triangulation, which both are proven. The quality guarantee and the size-optimality do exist for the three-dimensional generalization. The main purpose for the use of the grid is to control the quality of the elements. The subdivisions of the grid are controlled by the geometry of the problem.

In this algorithm, the quadrant will be subdivided if the quadrant is crowded. Figure 1.16 shows some examples of a crowded quadrant. By definition, the quadrant  $q$ , whose edge length is  $l$ , is crowded if  $q$  contains a vertex  $p$  from the domain  $G$  and one or more of the following conditions are met:

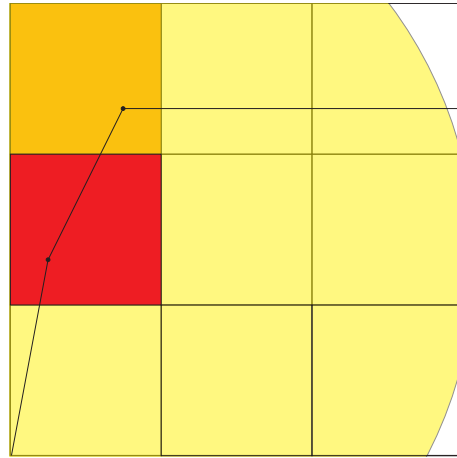
- $q$  contains two or more vertices from  $G$ .
- A vertex from  $G$  is within  $2\sqrt{2}l$  units of  $p$ .
- An extended neighbor  $q$  is subdivided.

An extended neighbor is an adjacent quadrant that of the same size. The extended neighbor also include quadrants of the same size that share a corner.

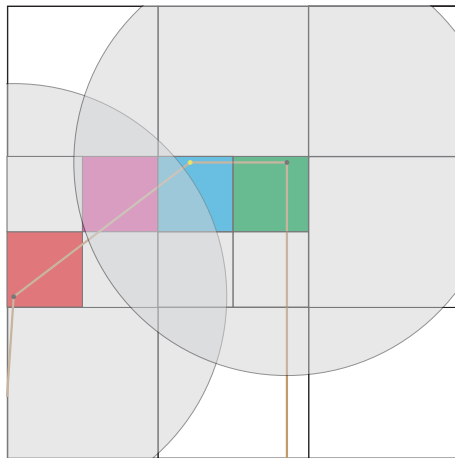
In Figure 1.16(a), the quadrant that is colored in red is crowded because the quadrant contains two vertices from the domain. In Figure 1.16(b), the yellow area is the interior of a circle with the radius  $2\sqrt{2}l$ , where  $l$  is the length of the red quadrant, units and the vertex in the red quadrant as its center. The vertex in the orange quadrant is within  $2\sqrt{2}l$  from the vertex in the red quadrant because it is inside the yellow area. Therefore, the red quadrant is crowded. In Figure 1.16(c), the figure on the left is the grid before inserting the yellow vertex, and the figure on the right is the grid after inserting the yellow vertex (but before subdividing the red quadrant). The gray disks on the left figure have a radius  $2\sqrt{2}l$  respectively to the quadrant that contains the corresponding center. The magenta quadrant is an extended neighbor of the red quadrant. After inserting the yellow vertex, the two blue and green quadrants are subdivided because they are crowded for being less than  $2\sqrt{2}l$  apart. Afterwards, the orange and magenta quadrants are subdivided to



(a) The crowded quadrant contains two vertices from the domain.



(b) The yellow disk has a radius of  $2\sqrt{2}l$  units with the vertex in the crowded box as its center. The vertex in the orange quadrant is within  $2\sqrt{2}l$  of the vertex in the red quadrant because the yellow disk contains the vertex in the orange quadrant in its interior.



(c) Grid before inserting the yellow vertex is on the left. Grid after inserting the yellow vertex is on the right. Extended neighbor is subdivided. Thus, red quadrant is crowded.

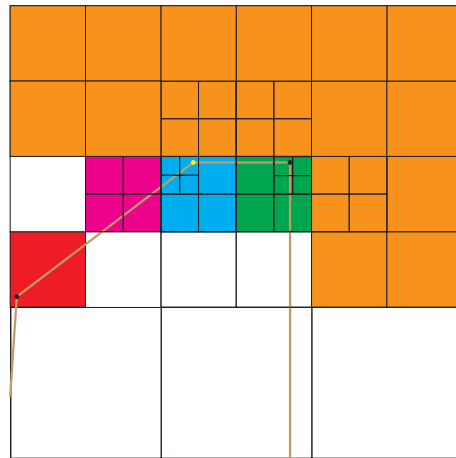


Figure 1.16: Crowded quadrants. Each example violates one of the three conditions. The crowded are colored in red. (a) Two vertices in a quadrant. (b) Vertices are within  $2\sqrt{2}l$  where  $l$  is the [larger] length of the quadrant they are in. (c) An extended neighbor is subdivided.

enforce the 2:1 balance condition, which forbids adjacent quadrants to be more than twice as large or less than twice as small. Since the magenta quadrant is subdivided, the red quadrant becomes crowded.

When the grid is subdivided fine enough, the grid will warp itself to the domain. If a quadrant contains a vertex, the closest corner will merge to that vertex. Figure 1.17 demonstrates the grid being merged to the vertices of the domain, and Figure 1.18 illustrates an example of the grid warping to the domain. After the warping process, the quadrants are triangulated using a constrained Delaunay triangulation algorithm.

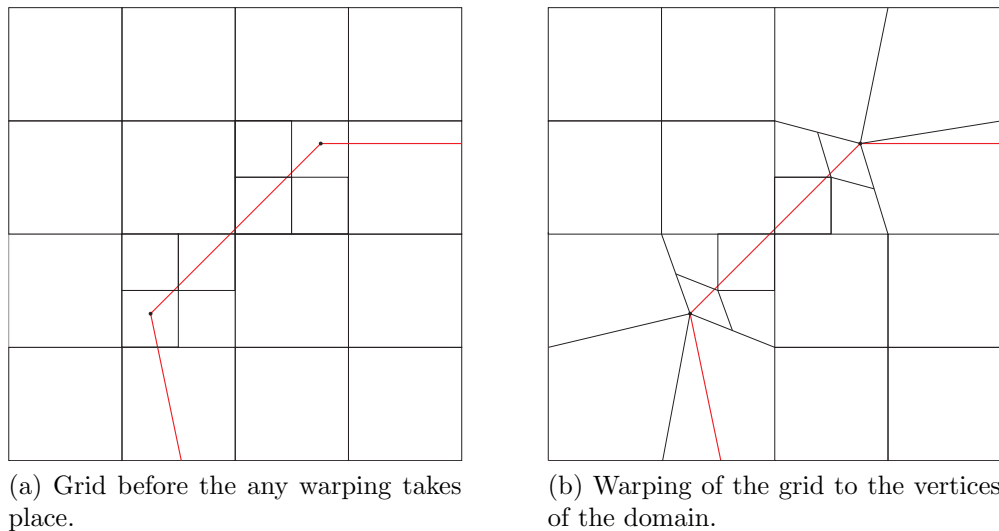


Figure 1.17: Warping of the grid to vertices of the domain.

The work of Bern, Eppstein, and Gilbert guarantees all angles are greater than  $18.4^\circ$  given that there are no angles that are smaller than  $18.4^\circ$  in the geometric model. This algorithm also produces a size-optimal mesh. However, the resulting mesh size is very large in practice.

Schroeder had done previous work on a combined octree/Delaunay algorithm [24]. This is a three dimensional algorithm, and it can be generalized to two dimensions. His algorithm reads in a geometric domain, mesh control parameter, and merge tol-

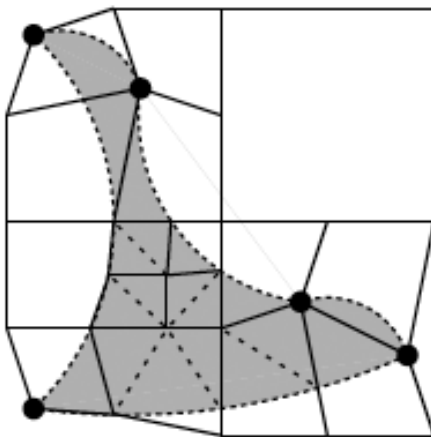


Figure 1.18: Warping of grid to the edges of the domain [9].

erance. Unlike the work of Bern, Eppstein, and Gilbert, the grid subdivides to the user defined mesh control parameter. The subdivision does not depend on the model except in the case where the mesh of the model is topologically inconsistent with the original model. The function of the grid in this work is to allow the method to be run in parallel.

Additionally, his work also considers merge operations. Unlike the warping in the work of Bern, Eppstein, and Gilbert, merging occurs only when two vertices are within the merge tolerance. No mathematical guarantees are proven or stated in his thesis. It is mentioned that the merging process will eliminate some poor quality elements.

Mitchell and Vavasis developed an algorithm [25] in this category that also offers quality guarantees and size-optimal triangulation. Similarly to the algorithm of Bern, Eppstein, and Gilbert, the mesh size in practice is very large. However, not all grid-based mesh generations offer guarantees. Shephard and Georges [26] offer no bounds but produce a smaller mesh size in practice than the work of Bern, Eppstein, and Gilbert. However, they use a smoothing algorithm to eliminate poor quality elements.

The advantages of grid and quadtree algorithms are their parallelism and mathematical guarantees. They are very robust. They can be designed to handle geometries that are created from a poor quality CAD tool. Of the three categories, algorithms



in this category are the only ones that can mesh a model using a top-down approach. However, they generate a large mesh size in practice.

## 1.5 Contributions

In this research, the 2-dimensional combined quadtree/Delaunay method based on Schroeder's combined octree/Delaunay algorithm [24] will be implemented. The following work will provide the following features:

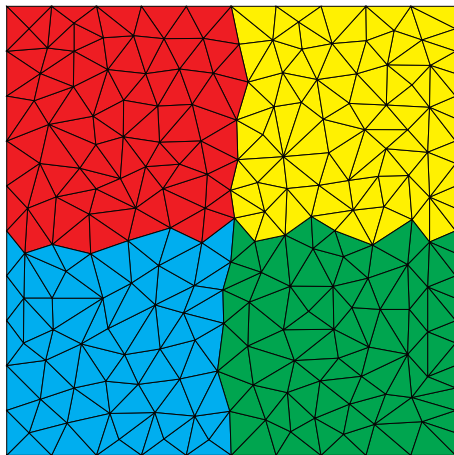
- Control the mesh size through the mesh control parameter
- Subdomains have flat interfaces
- Control element merging likelihood through merge tolerance
- Work with parametric curves and non-manifold geometries
- Fully automated

This work allows the users to control the size and number of elements through the mesh control parameters. Higher mesh control parameters yield a better approximation of the domain but a greater number of elements and a longer run time to obtain a mesh. The number of elements and the element size are inversely proportional. In some computational problems, it may be necessary to have small element size to capture the effects of rapidly changing solutions. Finally, a greater number of elements may increase the numerical accuracy of mesh-based scientific computing methods but requires more memory and run time.

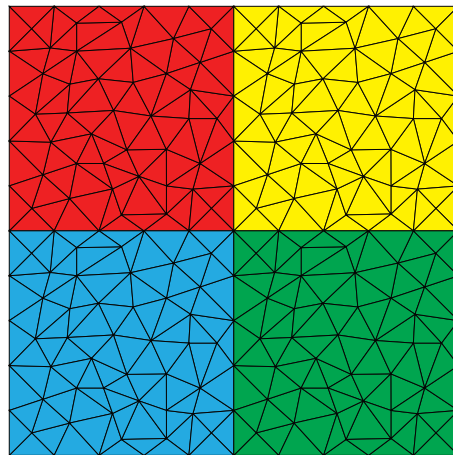
The boundaries of the subdomains in this approach are flat as shown in Figure 1.19(b). This property contributes to a finite element method called domain decomposition. Domain decomposition solves the boundary value problem on each meshed subdomain. Jagged subdomains, as shown in Figure 1.19(a), is computationally difficult to solve due to the boundaries of the subdomains. Using this method,

the subdomains are [square] quadrants, and the domains are easier to solve because boundaries are flat.

Like the size and number of elements, the users can control the number of subdomains through the mesh control parameters. Increasing the mesh control parameters will increase the number of subdomains. If the user have access to many parallel machines, this method can benefit greatly from having more subdomains.



(a) Four domains are assigned different colors: red, blue, green, and yellow. The interfaces that are shared between two domains are jagged.



(b) Four domains are assigned the different colors: red, blue, green and yellow. The interfaces that are shared between two domains are flat.

Figure 1.19: A domain with jagged interfaces and a domain with flat interfaces.

This approach allows the users to control over the merge tolerance. Higher merge tolerance yields better mesh quality by eliminating some poor quality elements. However, eliminating some of these elements may lead to a less accurate approximation of the domain.

This approach will work on non-manifold shapes with parametric curves, such as a circle. By allowing parametric curves, the users may be able to model geometries more accurately. In addition, it may lead to a better mesh approximation.

Finally, this approach is fully automated. Once the model of the domain is created and its attributes (mesh control parameters and merge tolerances) are assigned, the algorithm will terminate without requiring any more user input during its execution.

# CHAPTER 2

## APPROACH

### 2.1 Algorithm

The combined quadtree/Delaunay method can be divided into three components, spatial partitioning, successive triangulation, and mesh quality improvement. The spatial partitioning subdivide geometric domain into subdomains with square grid cells; this is illustrated in Figure 2.1(a). The successive triangulation follows after the spatial partitioning. The successive triangulation independently triangulates each of the grid cell as shown in Figure 2.1(b). At the final stage of the algorithm, the mesh quality improvement procedure that involves merging of geometry intersection vertices to grid vertices is to eliminate poor quality elements from the mesh. Each step of the method will be discussed in detail in the following sections.

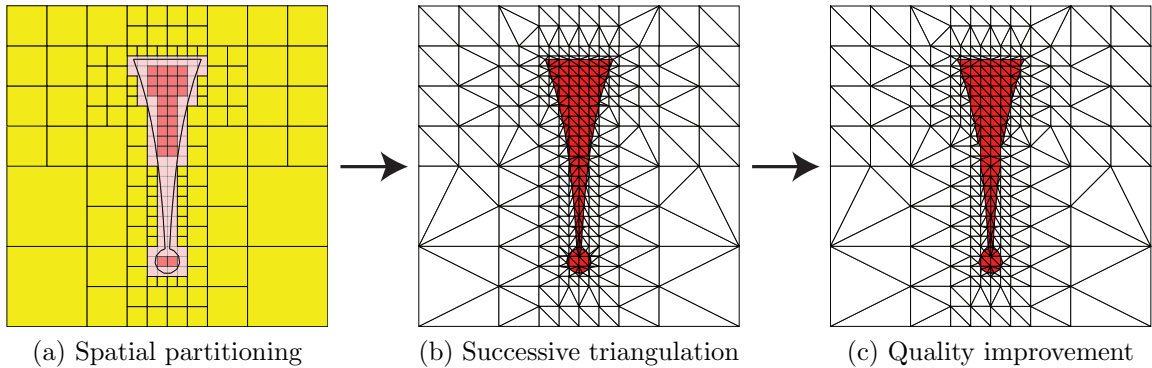


Figure 2.1: Overview of the proposed combined quadtree/Delaunay triangulation. (a) Space partitioning of the geometric model, and classification of quadrants (interior/boundary/exterior). (b) Independent constrained DT triangulation of each quadrant. (c) Elimination of poor quality triangles by merging select quadtree/geometry intersection point into the quadtree vertices.

### 2.1.1 Spatial Partitioning via Tree Data Structure

The purpose of the spatial partitioning component is to divide the geometric model into smaller and topologically simpler geometric subdomains that can be meshed independently and more easily. Additionally, the spatial partitioning component control the element mesh size according to the mesh control parameter.

There are three parts to spatial partitioning: inserting vertices, inserting edges, and inserting faces. Before these three processes, a root quadrant, or bounding box, is constructed as shown in Figure 2.2.

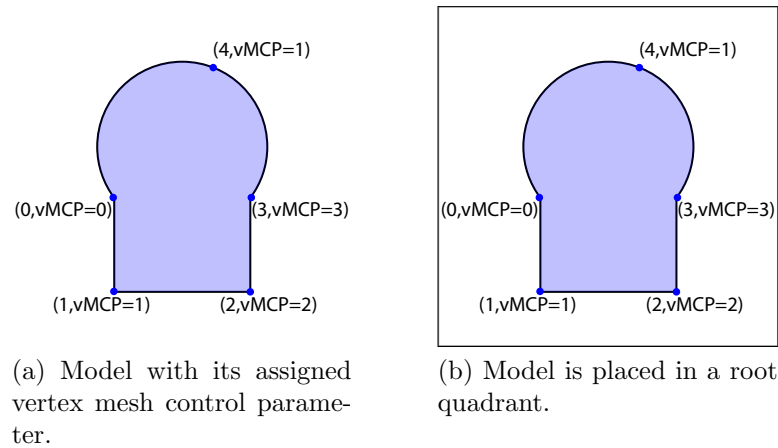


Figure 2.2: Geometric model with attributes and the respective root quadrant in the spatial partitioning. The vertex mesh control parameter (vMCP) is a mesh control parameter assigned to a vertex by the user.

#### 2.1.1.1 Inserting Vertices

The first third of the spatial partitioning procedure is the insertion of the vertices. This algorithm is described in Algorithm 1. At each point increment, thus the loop over  $V$  in Algorithm 1, a vertex is inserted into a terminal quadrant, or a quadrant that is not yet subdivided, that has a depth greater than or equal to the mesh control parameter of the inserted vertex (thus the if-condition in the algorithm). If none exist, then the algorithm will subdivide the terminal quadrant that currently contains the inserted vertex until there is one.

Partition( $q_r, V, Q, E$ )

INPUT:

$q_r$ : Root quadrant, or bounding box, that contains the model

$V$ : Set of topological vertices in the model

$Q$ : Set of quadrants, initially  $\{\}$

$E$ : Set of topological edges in the model

DEFINITIONS:

mcp( $t$ ): Returns the mesh control parameter of the topological element  $t$

children( $q$ ): Returns an empty set or a set of four child quadrants of the quadrant  $q$

depth( $q$ ): Returns the depth of the quadrant  $q$

vertex( $q$ ): Returns the set of inserted vertices that are interior to the quadrant  $q$

EnforceBC( $q$ ): See Algorithm 3

Subdivide( $q$ ): See Algorithm 2

ALGORITHM:

**for**  $v \in V$  **do**

**if** mcp( $v$ ) > depth( $q_r$ ) **then**

    Subdivide( $q_r$ , depth( $q_r$ ) + 1,  $E$ )

**for**  $c \in \text{children}(q_r)$  **do**

      Partition( $c, \{v\}, Q, E$ )

$Q = Q \cup \{c\}$

**end for**

**return**

**else**

    vertex( $q_r$ ) = vertex( $q_r$ )  $\cup$   $\{v\}$

**end if**

**end for**

**Algorithm 1:** Partitions model into quadrants by its vertices.

During the subdivision process, a parent quadrant will be subdivided into 4 equal child quadrants (if it is not already subdivided). Geometric vertices that are contained in each parent quadrant will be redistributed to the four child quadrants appropriately. Intersection between geometric edges and the quadrants will be computed to determine the shared vertices between adjacent quadrants. Determining the shared vertices here will localize the stitching process in the successive triangulation to each quadrant and avoid tolerance problems. The subdivision process is described in Algorithm 2.

Additionally, after each subdivision, the 2:1 balance condition is enforced. The 2:1 balance condition will be discussed in details in a later section. Any quadrant that has adjacent quadrants whose depths are less than its parent's depth will be subdivided. The 2:1 balance condition is described in Algorithm 3.

Subdivide( $q, d, E$ )

INPUT:

- $q$ : Quadrant to be subdivided
- $d$ : Depth of the terminal quadrants under  $q$  needs to be after subdivision
- $E$ : Set of topological edges in the model

DEFINITIONS:

- children( $q$ ): Returns an empty set or a set of four child quadrants of the quadrant  $q$
- depth( $q$ ): Returns the depth of the quadrant  $q$

ALGORITHM:

- 1: **If**  $d \leq \text{depth}(q)$ , **then** return.
- 2: **If** children( $q$ ) =  $\{\}$ 
  - i. Create two bisecting edges,  $b_1$  and  $b_2$ , of  $q$ .
  - ii. Cut  $q$  into four quadrants  $q_1, q_2, q_3, q_4$  with the two bisecting edges.
  - iii. children( $q$ ) =  $\{q_1, q_2, q_3, q_4\}$
  - iv.  $\forall i \in \{1, 2, 3, 4\} \text{depth}(q_i) = \text{depth}(q) + 1$
  - v. Redistribute  $v \in \text{vertex}(q)$  to  $q_1, q_2, q_3$ , or  $q_4$  that contains  $v$ .
  - iv. Distribute each intersection  $p$  between  $e$  and  $b_1$  and between  $e$  and  $b_2$  to  $q_1, q_2, q_3$ , or  $q_4$  that contains  $p$  [on its perimeter].
  - vii.  $\forall i \in \{1, 2, 3, 4\} \text{EnforceBC}(q_i, Q, E)$
- 3: **Else, then** call Subdivide( $c, d, E$ ) for each  $c \in \text{children}(q)$ .

**Algorithm 2:** Subdivides the quadrant.

```

EnforceBC( $q, Q, E$ )
  INPUT:
   $q$ : Quadrant that will be inspected and forced to meet the 2:1 balance condition
   $Q$ : Set of quadrants
   $E$ : Set of topological edges in the model

  DEFINITIONS:
  vertex( $q$ ): Returns the set of inserted vertices that are interior to the quadrant  $q$ 
  adjacent( $q$ ): Returns a set of terminal quadrants that are in  $Q$  and intersect with
   $p$  on an edge that has a non-zero length.
  depth( $q$ ): Returns the depth of the quadrant  $q$ 

  ALGORITHM:
  for  $q' \in \text{adjacent}(q)$  do
    Subdivide( $q', \text{depth}(q) - 1, E$ )
  end for

```

**Algorithm 3:** Enforces the 2:1 balance condition.

Figure 2.3 continues off from Figure 2.2 and demonstrates the walkthrough of this third of the spatial partitioning process on a concrete example. After constructing the root quadrant, the spatial partitioning can start inserting vertices. The spatial partitioning procedure will start with Node 0. Node 0 is inserted into the terminal quadrant, which is the root quadrant in this step. No subdivision is necessary because the depth of the root quadrant, 0, is equal to the mesh control parameter of Node 0 and the 2:1 balance condition is not violated. This step is shown in Figure 2.2(a). This algorithm continues with inserting the next node, Node 1. Node 1 is inserted into the root quadrant because the root quadrant is a terminal quadrant that contains Node 1. The root quadrant must be subdivided because the mesh control parameter of Node 1, 1, is greater than its depth, 0. During the subdivision of the root, the geometric edges intersect the two bisecting edge that subdivides the root quadrant to determine the shared vertices on the shared interfaces between two quadrants. The vertices that were insert into the root quadrant, Nodes 0 and 1, are redistributed to the corresponding child quadrants that contain them. In this case, both vertices are redistributed to the bottom left quadrant. After the subdivision, the quadrant



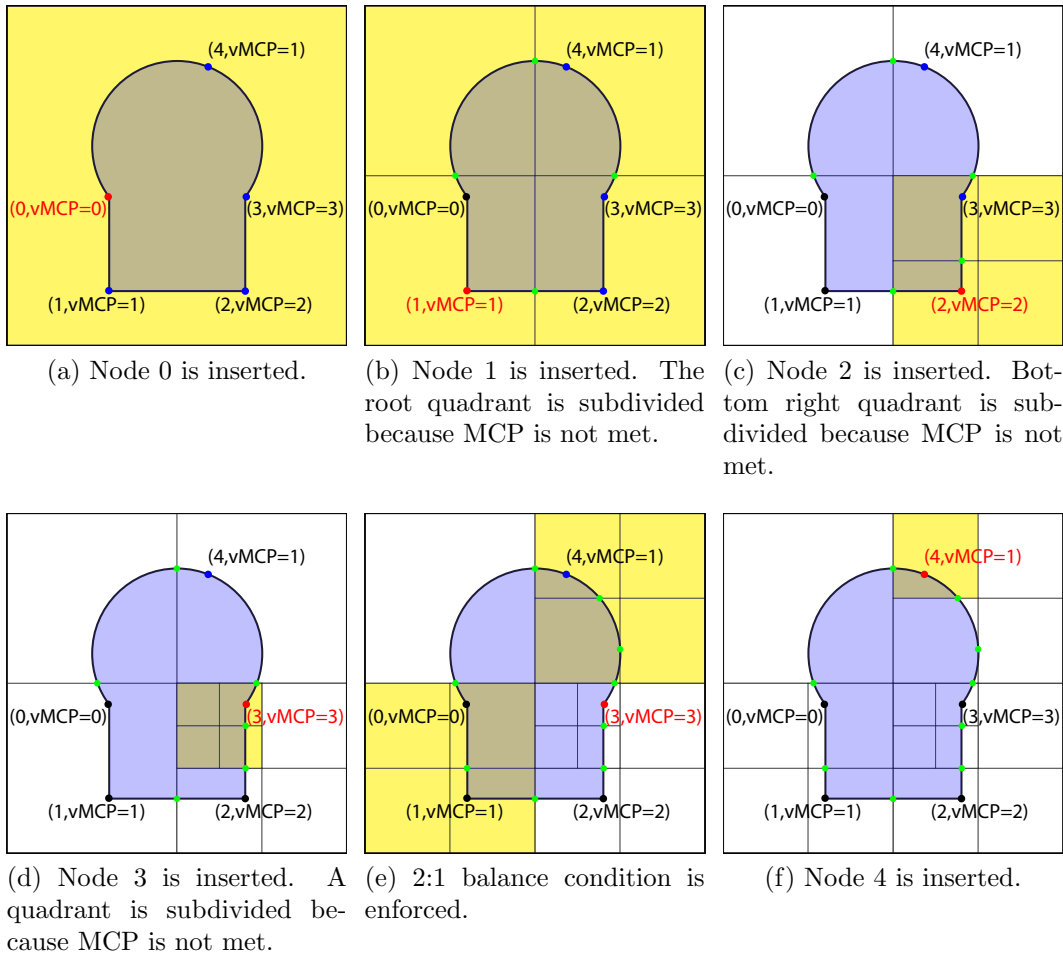


Figure 2.3: Insertion of geometric vertices, quadrant subdivision, computation of the corresponding intersections, and 2:1 balance condition.

that contains Node 1, the depth of the bottom left quadrant, is equal to the mesh control parameter of Node 1. The 2:1 balance condition is not violated yet. Therefore, the algorithm is finished with Node 1, and the figure is shown in Figure 2.2(b). The algorithm then continues to the next node, Node 2. The terminal quadrant that contains Node 2 is the bottom right quadrant. This quadrant must be subdivided because the mesh control parameter of Node 2, 2, is greater than its depth, 1. Like the previous subdivision, intersections are performed and vertices are redistributed. After the subdivision, the mesh control parameter of Node 3 is equal to the depth of the terminal quadrant that contains it. Like the rest, Node 3 is inserted to its terminal quadrant. That terminal quadrant is subdivided because the depth is smaller than the mesh control parameter. After the subdivision (including the redistribution of vertices and intersections), the smaller quadrants violate the 2:1 balance condition. Therefore, the 2:1 balance condition is enforced by subdividing the larger adjacent quadrants, which are the bottom left and top right quadrants. The bottom left quadrant has vertices in it. Therefore, during its subdivision, it will redistribute its vertices to the smaller quadrants. Like always, intersections take place in both of these quadrants. After subdivision of these two quadrants, the subdivided quadrants meet the 2:1 balance condition. Returning to meeting the mesh control parameter of Node 3, the terminal quadrant that now contains Node 3 has a depth that equals to the node's mesh control parameter. All quadrants meet the 2:1 balance condition, and the algorithm can continue to the next node. Node 4 is inserted like the previous nodes. After inserting Node 4, the algorithm finishes with inserting all of the vertices in domain, and the algorithm will move onto inserting edges, which will be discussed in the sections ahead.

### 2.1.1.2 Mesh Control Parameter

The mesh control parameter (MCP) is a parameter that the user assigns to a topological element. It controls the maximum size of the grid cells that contain the topological element. Alternatively, the mesh control parameter control the number of subdivisions the quadrant that contain the topological element must undergo. The mesh control parameter can be specified by its unit length (maximum length or width of the quadrant) or the number of subdivisions (depth in the tree data structure). In this thesis, the mesh control parameter will be specified by its depth. The conversion from unit length to its depth is found in (2.1) where  $L_r$  is the length of the starting quadrant that contains the whole domain and  $L_{mcp}$  is the specified mesh control parameter in unit length.

$$\text{MCP} = \left\lceil \log_2 \left( \frac{L_r}{L_{mcp}} \right) \right\rceil \quad (2.1)$$

where  $\lceil \cdot \rceil$  operator is the ceiling operator that rounds up a real number to an integer.

As mentioned in the introduction, the mesh control parameter is topological attribute. The mesh control parameter only affect the quadrants which contain or partially contain its topological entity, namely vertex (vMCP), edge (eMCP) or face (fMCP). This is illustrated in Figure 2.4.

The mesh control parameter is 0 by default if one is not assigned to a topological entity. Additionally, if the mesh control parameter of a smaller dimensional topological entity is less than the mesh control parameter of its associated greater dimensional topological entity, then the smaller dimensional topological entity will inherit the mesh control parameter of its associated greater dimensional topological entity. For example, if the assigned mesh control parameter of an edge  $E_i$  is  $eMCP_i = 2$ , and the assigned mesh control parameter of its incident face  $F_j$  is  $fMCP_j = 5$ , then the mesh control parameter of  $E_i$  is  $eMCP_i = 5$ .

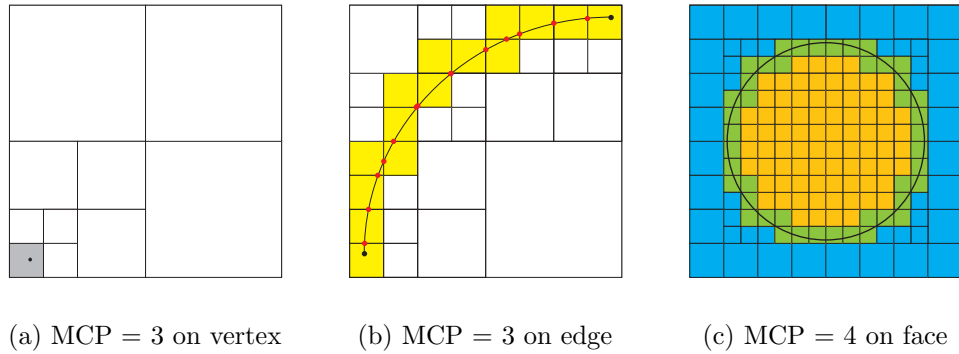


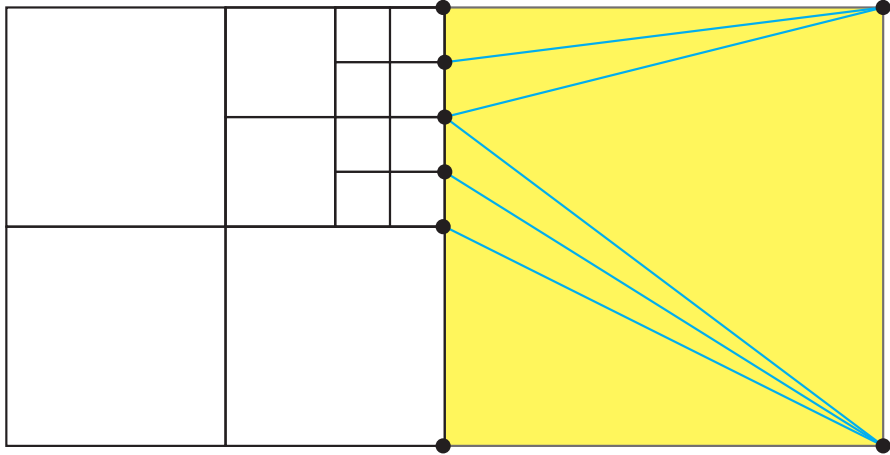
Figure 2.4: Mesh control parameter (MCP) on different topological entities. (a) Vertex. (b) Edge. (c) Face.

### 2.1.1.3 2:1 Balance Condition

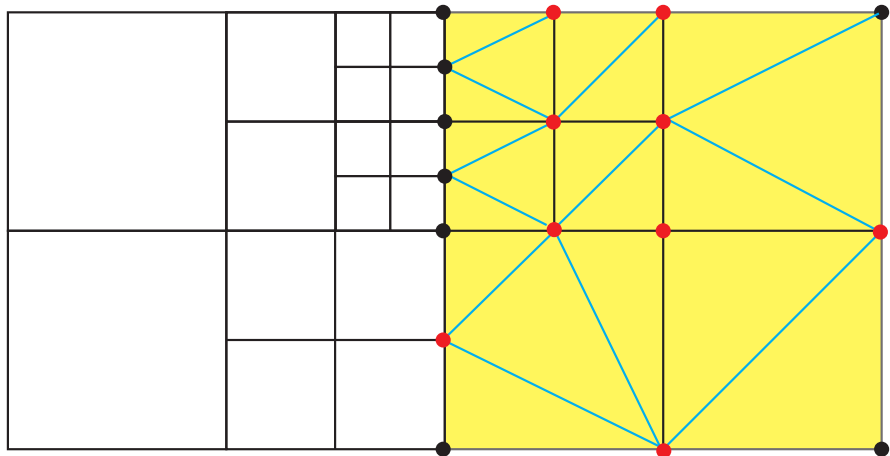
The 2:1 balance condition ensures that the edge length of a quadrant is no more than twice as large or no less than twice as small as the length of any adjacent quadrants. The purpose of the 2:1 balance condition is to eliminate bad quality elements from smaller adjacent quadrants as shown in Figure 2.5(a). Figure 2.5(b) shows the improvement in element quality if the 2:1 balance condition is enforced.

### 2.1.1.4 Inserting Edges

The second third of the spatial partitioning process is the insertion of edges. During this part of the process, the algorithm enforces that the depths of the quadrants that an edge passes is greater than or equal to the mesh control parameter of that edge and identifies the segments that need to be preserved for the successive triangulation process. The quadrants that contain the edge can be retrieved from the intersections with the grid and the two end points. If a retrieved quadrant does not meet the mesh control parameter of that edge, then that retrieved quadrant will be subdivided. This continues until all of the retrieved terminal quadrants meet the mesh control parameter of the edge. Figure 2.6 illustrates this process.

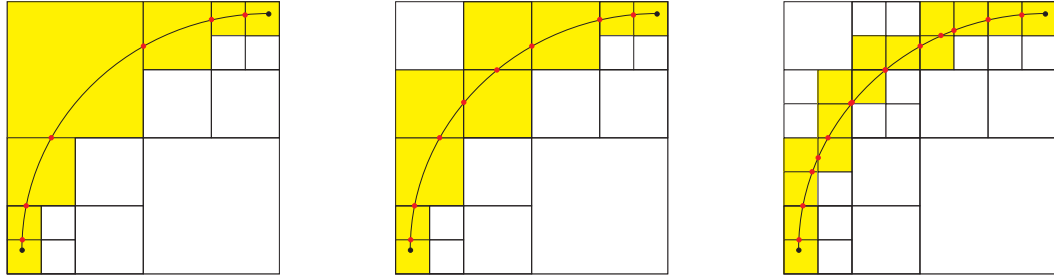


(a) A possible scenario where 2:1 balance condition between quadrants is violated (in this case the ratio of the largest to the smallest neighboring quadrant edges are 4:1). The highlighted quadrant violates the 2:1 balance condition, and its successive triangulation creates four skinny triangles.



(b) This figure is the same scenario as Figure (a), but 2:1 balance condition between quadrants is enforced.

Figure 2.5: Benefits from enforcing the 2:1 balancing condition on the quadtree spatial partitioning. Additional vertices that are added due to the 2:1 balance condition are colored in red.



(a) Obtain the terminal quadrants that contains the geometric edge. These quadrants are colored in yellow.

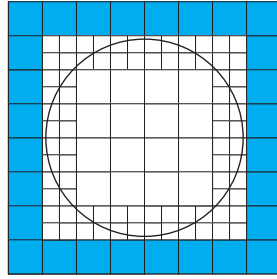
(b) Some quadrants are subdivided because they do not meet the MCP of the edge. Continue subdividing until the quadrants that contain the edge meet the MCP.

(c) All of the highlighted quadrants meet the mesh control parameter of the edge.

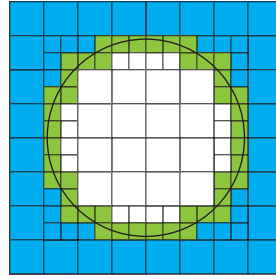
Figure 2.6: Insertion of an edge (curve) with  $eMCP=3$  into the quadtree space partitioning. Note that while the quadrants are refined, the new quadrants must be re-classified as boundary (yellow).

### 2.1.1.5 Inserting Faces

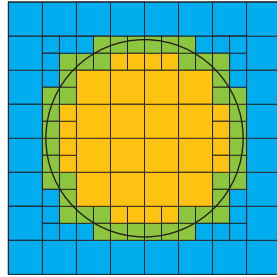
The final third, and final step, of the spatial partitioning process is the insertion of faces. This part of the process enforces that the depths of the quadrants contain a geometric face greater than or equal to the user-defined mesh control parameter of that face. This part of the process starts off by classifying all of the empty quadrants on the perimeter of the bounding box as exterior quadrants. The algorithm then sweeps the grid for empty quadrants and classifies them as exterior quadrants. The quadrants that contain edges are classified as boundary quadrants. The algorithm then sweeps the grid, starting from previously encountered boundary quadrants, and classifies the swept empty quadrants as interior quadrants. The interior quadrants are then recursively subdivided until its depth is greater than or equal to the greatest mesh control parameter of the faces they contain. Figure 2.7 is a demonstration of this process on a geometric domain with a circular disk.



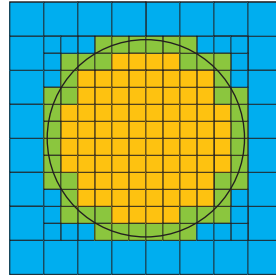
(a) The algorithm classifies the quadrants on the perimeter of the bounding box as exterior quadrants.



(b) The algorithm sweeps for exterior quadrants.



(c) The algorithm sweeps for interior quadrants.



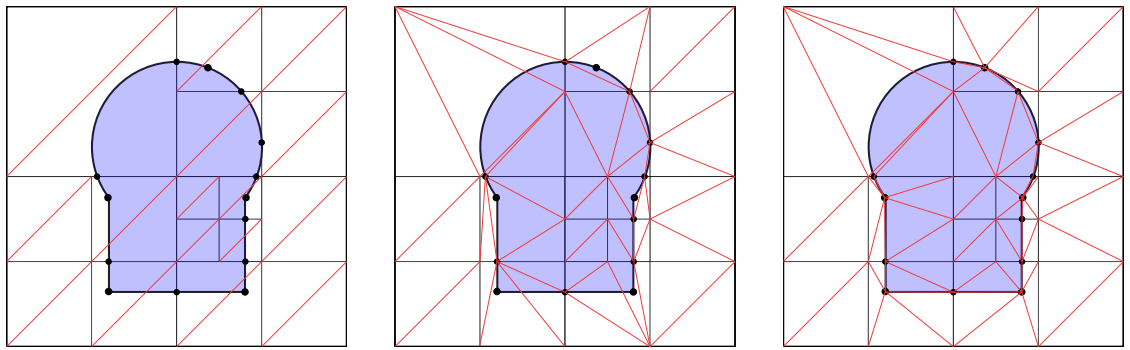
(d) The algorithm recursively subdivides interior quadrant to mesh control parameter of its contained face.

Figure 2.7: Insertion of a face (circular disc) with  $fMCP=4$  into the quadtree space partitioning. Note that while the quadrants are refined, they must be classified as interior, exterior and boundary quadrants.

### 2.1.2 Successive Triangulation

The purpose of the successive triangulation is to produce a triangulation in pieces. The separate triangulations as a whole form the triangulation of the domain. The successive triangulation procedure follows after the spatial partitioning procedure is finished. The successive triangulation starts with a triangulation of two isosceles right triangle as Figure 2.8(a). It then inserts all vertices as illustrated in Figures 2.8(b) and (c). Finally, the successive triangulation preserve the edges by stitching them as shown in Figure 2.8(d). The critical step in the successive triangulation process

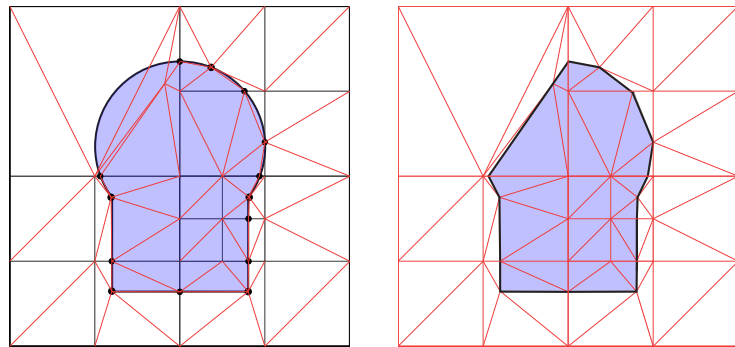
is to produce a globally consistent mesh while independently meshing the quadrants. This can be achieved with two different approaches: (a) having a conforming DT at each quadrant with no Steiner points at the quadrant boundaries; or (b) with a set of predefined Steiner points on quadrant edges, that will be used by the DT refinement algorithm of each neighboring quadrant. Each quadrant edge should be constrained to have no Steiner vertices on it.



(a) Each quadrant is triangulated using a template.

(b) The vertices on the perimeter of the quadrant are inserted.

(c) The vertices in the interior of the quadrant are inserted.



(d) The missing edges are stitched.

(e) Triangulation

Figure 2.8: Overview of the successive triangulation steps through an example



### 2.1.2.1 Conforming Delaunay Triangulation

Introducing Steiner points on the boundary will produce a similar triangulation shown in Figure 2.9. If quadrants are forced to match, then there will be some dependencies on adjacent quadrants. The conforming Delaunay triangulation will achieve this constraint by using the quadrant as the bounding box of the triangulation and inserting vertices on the boundary before any interior vertices.

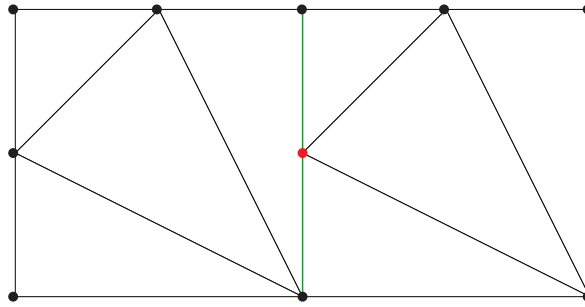


Figure 2.9: The right quadrant introduces the red point. The triangulation on left does not match the triangulation on the right on the interface, or the green edge.

In the successive triangulation process, the Delaunay triangulation, using Bowyer-Watson algorithm, is performed on the points that are local to the quadrant. The Delaunay triangulation starts with a triangulated template of a bounding box, defined by the corners of the quadrant. This triangulated template is shown in Figure 2.10. The vertices on the boundaries are inserted before the vertices that are interior to the quadrant. After inserting the interior vertices, the algorithm does a check for missing edges and preserves them. The algorithm stitches the missing edges by inserting points where the missing edges or segments intersect the triangulation. An example of the stitching of edges are shown in Figure 2.11.

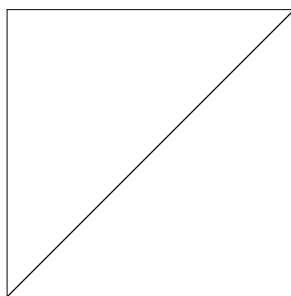


Figure 2.10: Triangulated template of a quadrant

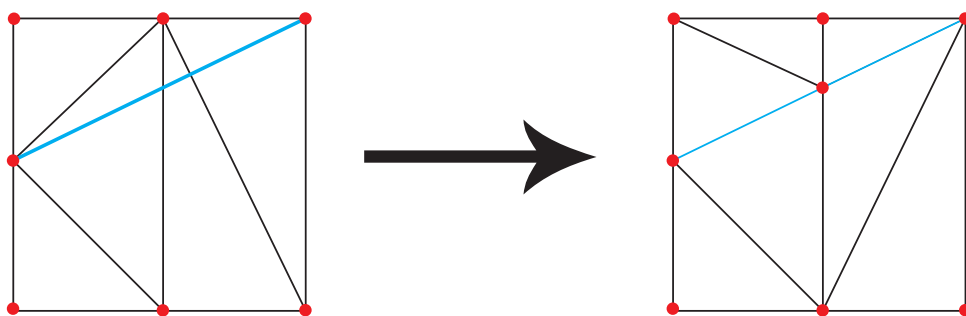


Figure 2.11: Preservation of geometric edges in a quadrant by stitching. The missing blue edge on the left figure is recovered on the right by recursively inserting the intersection vertex of the edge with the existing mesh edges via the Bowyer-Watson algorithm.

### 2.1.2.2 Delaunay Refinement

Any conforming triangulation method can be used as a successive triangulation method. Delaunay refinement algorithms produce conforming Delaunay triangulations. In addition, Delaunay refinement algorithms introduce Steiner points into the triangulation to improve element quality. Ruppert's Delaunay refinement algorithm is one of the earliest work in Delaunay refinement, and it is described in the introduction. In this section, the Delaunay refinement algorithm will be used as the successive triangulation method. Figure 2.12 shows an overview of this process. Before performing a Delaunay refinement algorithm on each quadrant, the algorithm needs to take

each quadrant from the spatial partitioning process and refine the perimeter of the quadrants.

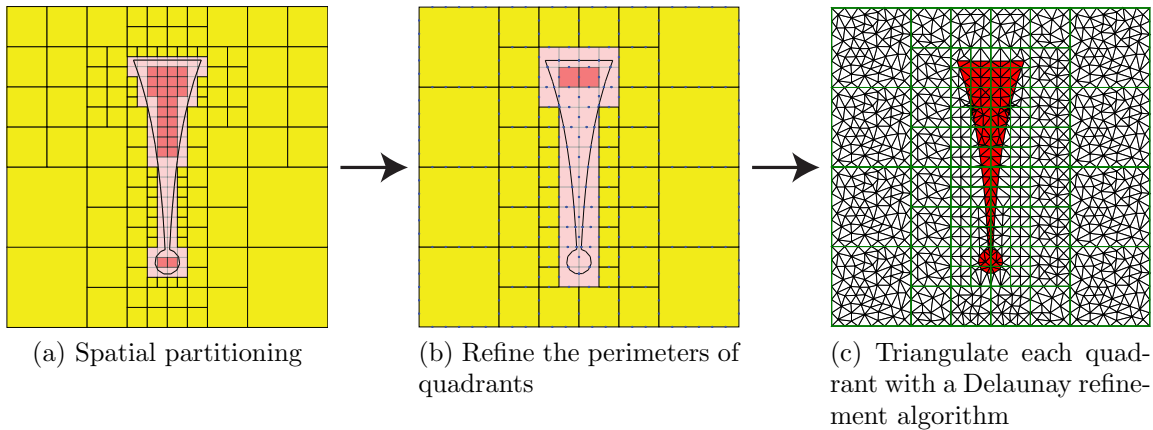


Figure 2.12: Overview of the successive triangulation method with a Delaunay refinement algorithm. (a) Space partitioning of the geometric model. (b) Refinement of the quadrant boundaries. (c) Delaunay refinement of each quadrant.

When using Delaunay refinement algorithms, it is important to control where Steiner points are inserted. Otherwise, there will be problems on the perimeter of the quadrant as shown in Figure 2.13. This is an example of nonconforming triangulation. A nonconforming triangulation is a triangulation with triangles that may have additional vertices on their edges. Figure 2.13 illustrates the three triangles that make this triangulation a nonconforming triangulation; they are colored in blue and orange.

This problem arise only when an upper bound on the size of element is specified in the Delaunay refinement algorithm. Without the upper bound on the size of element, the Delaunay refinement algorithm will return a conforming Delaunay triangulation that was used as a successive triangulation method earlier in this section.

To address this issue, the edges along the perimeter of the quadrant are constrained edges. These edges must be in the triangulation, and each edge cannot be represented by more than one edges from the mesh. However, more work is needed to produce a better quality triangulation. Figure 2.14 illustrates the poor quality elements that

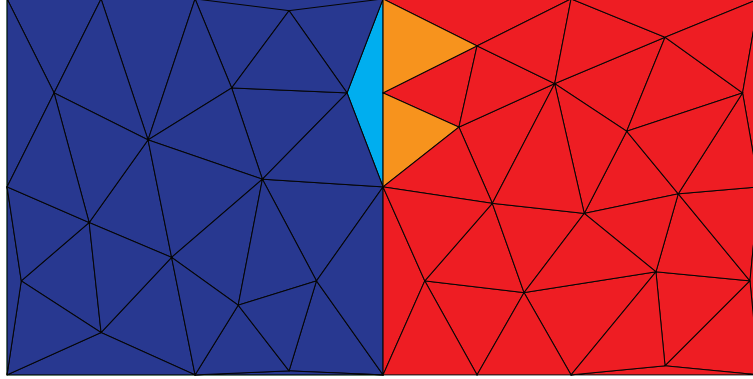


Figure 2.13: Possible non-conforming meshing between quadrants if the Delaunay refinement is independently applied on each neighboring quadrant.

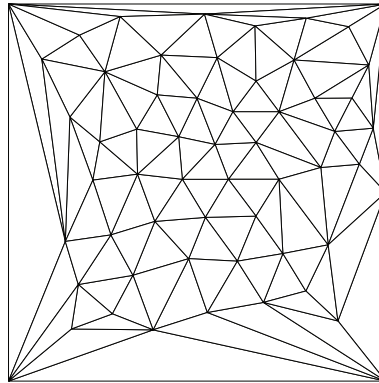


Figure 2.14: Triangulation of an empty quadrant with the edges on the perimeter of the quadrant as constrained edges.

are caused by only making the edges on the perimeter of the quadrant as constrained edges.

In the figures that are shown in this subsection, the triangulation is generated by using a Delaunay refinement algorithm with the maximum area as a criteria. To address the problem with the quality, the edges along the perimeter of the quadrant need to be subdivided until their lengths are less than:

$$l_t = \sqrt{\frac{4}{\sqrt{3}} A_{max}} \quad (2.2)$$

where  $A_{max}$  is the specified upper bound on the element area. The threshold length  $l_t$  in (2.2) has the same length as a side of an equilateral triangle with the area  $A_{max}$ . Figure 2.16 illustrates a triangulation with two discussed resolutions accommodated.

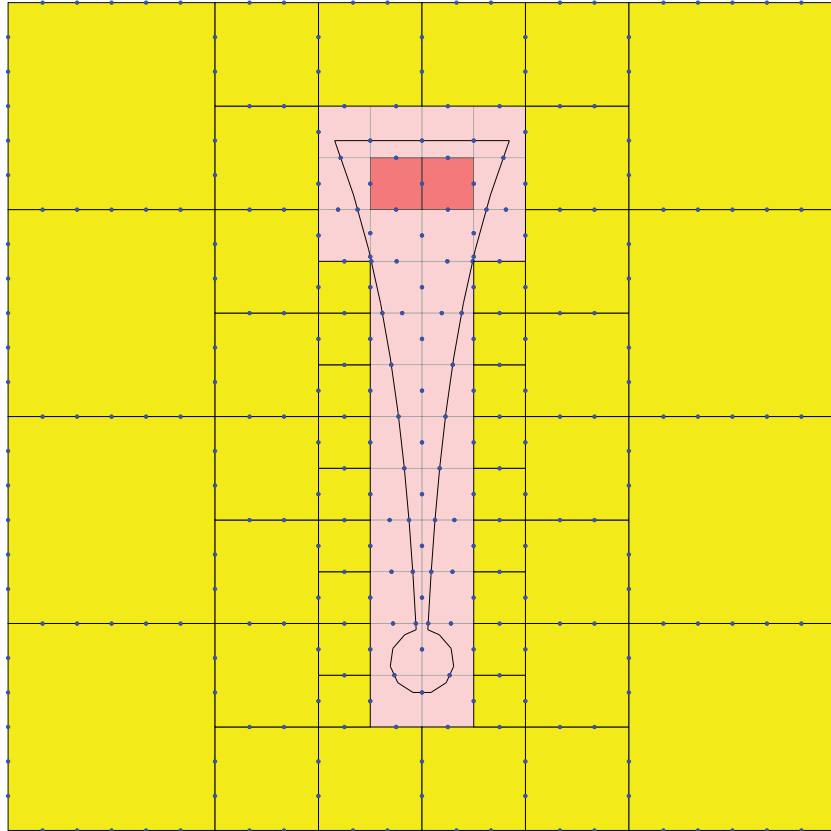


Figure 2.15: Refinement of the quadrant boundaries. Vertices on the quadrant boundaries are no more than the threshold length  $l_t$  apart as defined in (2.2).

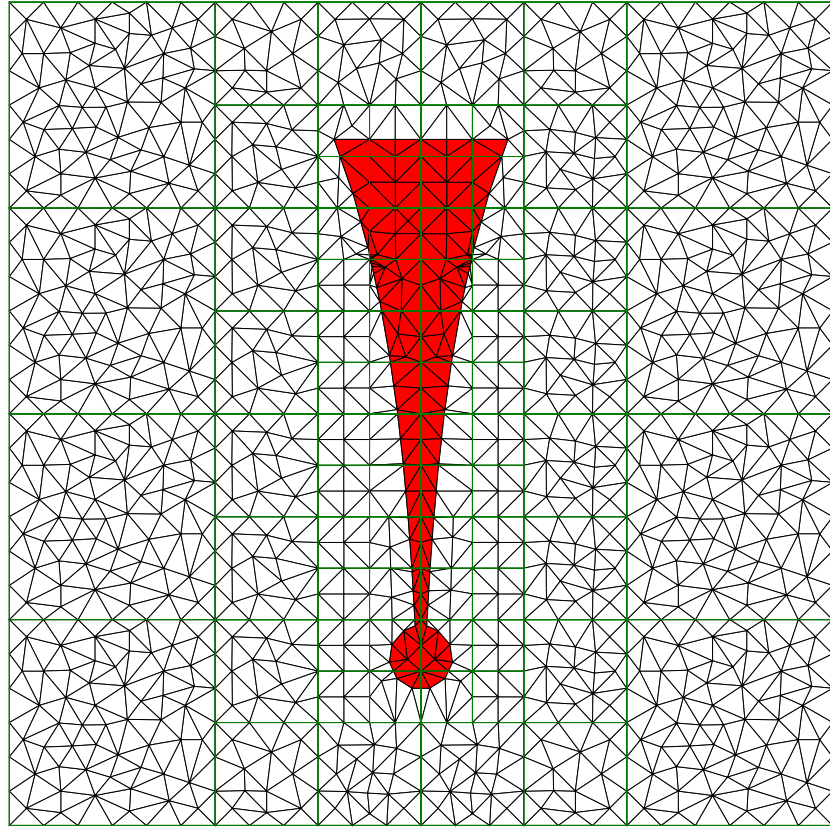


Figure 2.16: Combined quadtree and successive triangulation, where the triangulation is performed by the Delaunay refinement from the Triangle [16] code. Delaunay refinement algorithm as a successive triangulation method. The edges along the perimeter of the quadrant are set to be constrained edges and subdivided until it is smaller than the threshold length.

### 2.1.3 Mesh Quality Improvement

The purpose of mesh quality improvement module is to eliminate poor quality elements from the triangulation by merging nearby vertices or collapsing skinny triangles as shown in Figure 2.17. However, the approximation of the domain will be traded off for each merge operation. The merge tolerance controls the merging distance and is specified by the user at input along with the geometric domain and the mesh control parameters. The merge tolerance is an user-defined ratio between the quadrant’s length and the merge distance. The merge tolerance is one parameter for the whole domain and can be any real value between 0 and 0.5. The merging distance is defined as follow:

$$d_m = m_t \text{Length}(\text{SharedTreeElement}(u, v)) \quad (2.3)$$

where  $m_t$  is the merge tolerance. The “SharedTreeElement” function returns the edge of a quadrant or a quadrant which is shared by the two vertices  $u$  and  $v$ . The “Length” function measures the length of an edge or the length of any edge which is incident to the given quadrant.

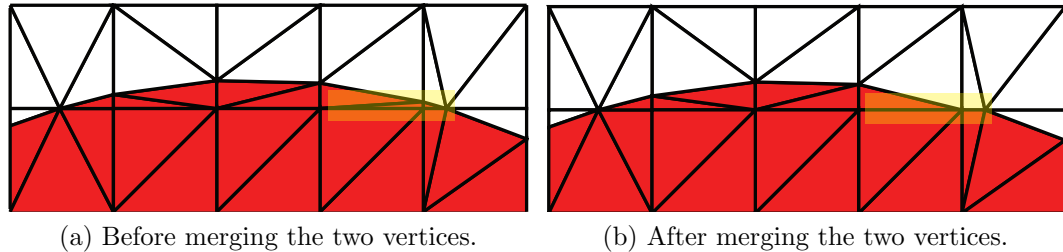
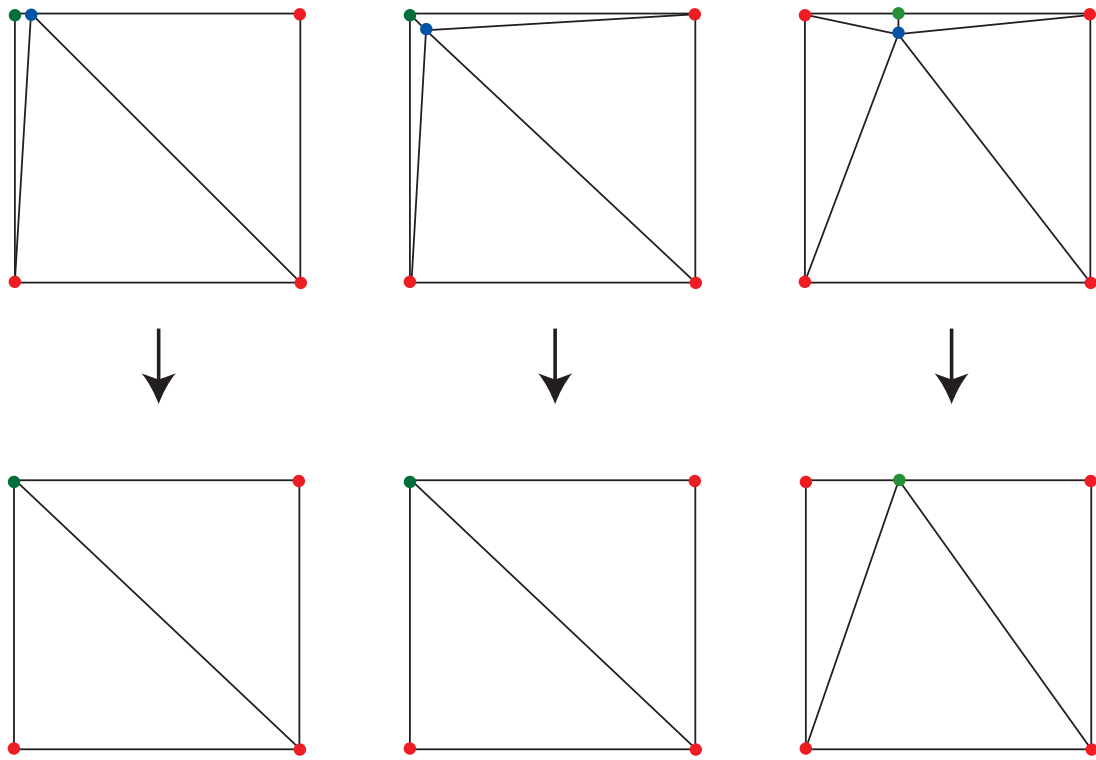


Figure 2.17: Elimination of poor quality (skinny) triangles by merging select vertex points from resulting from the intersection of the quadrant and geometry to a quadrant vertex.

There are three types of vertices in this algorithm: the corners of the quadrants, intersection points on the boundary of the quadrants, and the vertices of the domain. The corners of the quadrants are fixed, and they will never merge or warp to any

other vertices. If any of the vertices from the other two categories are within the merging distance from the corners, then these vertices will merge to the corners. If the vertices from the domain are within the merging distance from an intersection vertex on the boundary of the quadrant, then the vertices from the domain will merge to the intersection vertex. If a vertex from the domain is within a merging distance from a corner of a quadrant and an intersection vertex, the vertex will merge to the corner.



(a) Intersection vertex merges to corner vertex. (b) Domain vertex merges to corner vertex. (c) Domain vertex merges to intersection vertex.

Figure 2.18: Merge cases that is considered in this research. The top figures show the states before the merge process, and the bottom figures show the states after the merge process. The blue and green vertices are within the merging distance. The blue vertices are merged to the green vertices.



### 2.1.4 Quality Guarantees

Although there are no mathematical quality guarantees that is offered by merging nearby vertices in general, there are mathematical quality guarantees for a limited number of use cases that meet the following conditions:

#### Condition 2.1.1:

- Quadrants do not have more than two intersections;
- Quadrants do not have more than one intersection on the same edge;
- Quadrants do not contain vertices from the domain or vertices from preserving edges.

Only these quadrant that meet these conditions were considered because they have a lower bound on the quality under the merging conditions shown in Figure 2.18. The lower bound does not exist for quadrants with multiple intersections on a single edge because the intersection can be very close. If this distance approaches 0, then the quality of the element that incident on these two vertices approaches 0. This is illustrated in Figure 2.19.

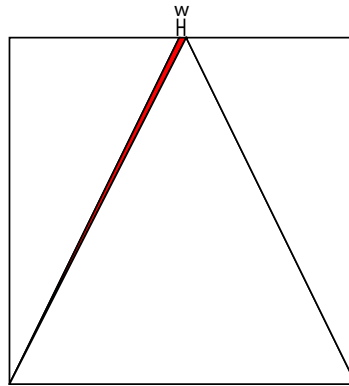


Figure 2.19: Case I, where the present merge strategy fails to filter poor quality triangles.  $\lim_{w \rightarrow 0} Q(\triangle) = 0$ , where  $\triangle$  is the red triangle and  $Q$  measures the quality of the triangle.

Additionally, the lower bound does not exist for quadrants that contain vertices from the domain. Figure 2.20 illustrates an example that shows there is no lower bound. It shows a vertex from the domain that is equidistant from the bottom-left and top-left corners of the quadrant. The merge tolerance is at the maximum, 0.5, and any vertices that is interior to the yellow circles are merged to the bottom-left corner or top-left corner. As the vertex approaches the midpoint of the edge that incident on the two corners, the quality of the red triangle approaches 0. The domain vertex will not merge to either corner as it approaches the midpoint because it is not within its merge distance. Therefore, there are no lower bound for quadrants that contain vertices from the domain.

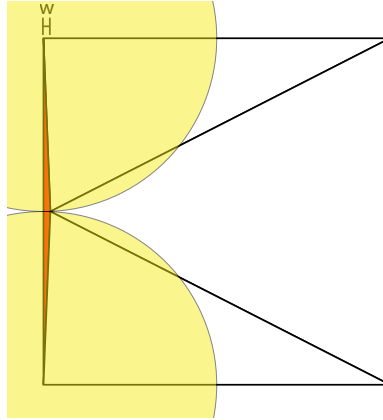


Figure 2.20: Case II, where the present merge strategy fails to filter poor quality triangles.  $\lim_{w \rightarrow 0} Q(\triangle) = 0$ , where  $\triangle$  is the red triangle and  $Q$  measures the quality of the triangle. The yellow areas marks the maximum merging distance of their corresponding centers.

Out of the quadrants that meet Condition 2.1.1, the assumed candidates for the quadrant with the poorest quality element are found in Figure 2.21. For each of the figure, its poorest quality element is colored red. Defining the merge tolerance to be  $m_t$  and the length of the quadrant to be 1 unit, the distance from each point on the edge to its closest corner is  $m_t$ . The computed qualities of these triangles are found in Table 2.1. The lower bound would be the minimum of these qualities at a

given merge tolerance. In the domain from 0 to 0.5, the expression in Table 2.1(a) will always yield the smallest value. Therefore, if all of the terminal quadrants meet Condition 2.1.1, the quality cut-off is:

$$q_t = \frac{m_t(1-m_t)\sqrt{3}}{2m_t^2 - m_t + 1}. \quad (2.4)$$

Figure 2.21	Computed Quality
(a)	$\frac{m_t(1-m_t)\sqrt{3}}{2m_t^2 - m_t + 1}$
(b)	$\frac{m_t(1-m_t)\sqrt{3}}{m_t^2 - 2m_t + 1}$
(c)	$\frac{m_t\sqrt{3}}{m_t^2 + 1}$
(d)	$\frac{m_t\sqrt{3}}{m_t^2 + 1}$
(e)	$\frac{m_t\sqrt{3}}{m_t^2 + 1}$

Table 2.1: The table computes the quality of the worst triangles of each triangulation in Figure 2.21.

## 2.2 Implementation

An implementation of this algorithm, for two-dimensional problems, will be deliverable at the end of the master thesis. The implementation will be written in C++ and uses the Open CASCADE framework [32]. Open CASCADE is a solid modeling framework, and it will be used to create an abstraction for intersection operations and modeling generic curves.

### 2.2.1 Data Structures

The data structures are divided into three categories: modeling, quadtree, and triangulation. Data structures in the modeling category will describe the model. As described in the introduction, a geometric model  $D = \{G, T, A\}$  is a collection of geometries  $G$ , topologies  $T$ , and attributes  $A$ . Data structures will be created for elements in the attributes domain and the topology domain. No data structures will

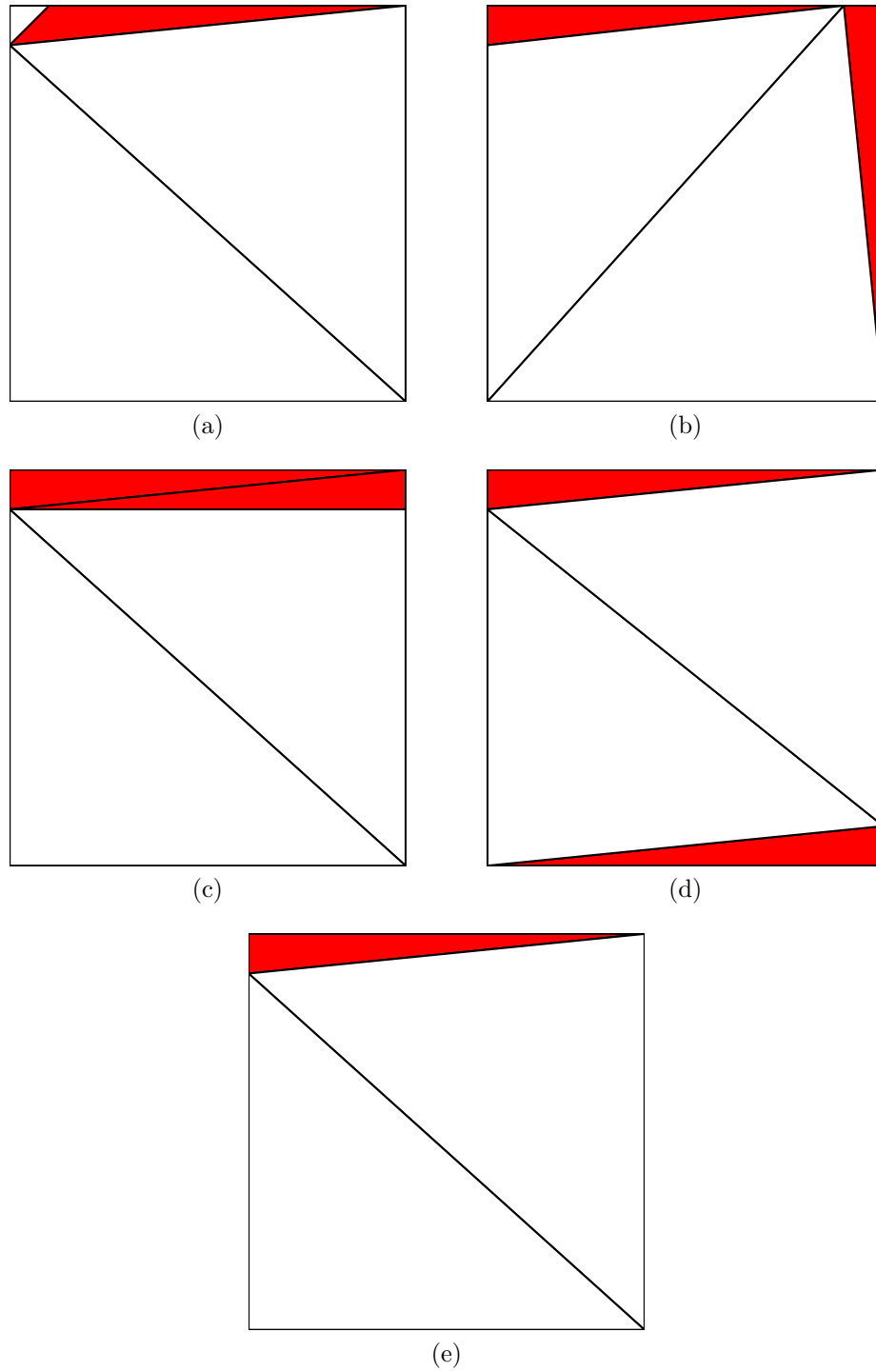


Figure 2.21: Merge candidate for triangles based on Condition 2.1.1. The red triangles will be removed to provide a low bound on the mesh quality.

be created for the geometry domain because this will be done by a third party library. Therefore, data structures that are created for modeling purposes are:

- Vertex - a 0-dimensional topological element that describes the topology of the model
- Edge - a 1-dimensional topological element that describes the topology of the model
- Face - a 2-dimensional topological element that describes the topology of the model
- Attributes - a collection of attribute elements
- AttributeElements - abstract data type for attribute elements.

Data structures in the quadtree category describe the bookkeeping that is needed in the spatial partition process. These data structures are:

- Corner - a corner of a quadrant. This is needed for the bookkeeping of vertices that are merged to the corner of the quadrants.
- Sectant - an edge of a quadrant. This is needed for the bookkeeping of the intersections with the grid and model. These intersections are generally shared by two quadrants. The successive triangulation for a quadrant will call this data structure to insert these intersections into the triangulation of the quadrant.
- Quadrant - a representation of a quadrant. It has a reference to the four child quadrants if it is subdivided. It also have one of the following classification: OUTSIDE, BOUNDARY, and INSIDE.
- IntersectionVertex - a point that is created from intersecting the geometric edge with the grid

- PreserveEdge - an edge that needs to be preserved in the triangulation of the quadrant. This is a first-order approximated, discretized edge of a geometric edge

Finally, the successive triangulation will need these data structures to represent the triangulation.

- MeshPoint - This is an inserted point in the triangulation.
- Triangle - This is a triangle from a triangulation.

### 2.2.2 Functions

The functions also divided into three categories: spatial partitioning, successive triangulation, and mesh quality improvement. The following functions are needed to perform the spatial partitioning component:

- intersect\_tree\_with\_domain ( 2 Sectants, Curve ) - intersects the sectant with the curve and stores the intersections in the corresponding halves of the sectants
- subdivide ( Quadrant, level ) - divides quadrant into four equal quadrants until the depth reach the specified level
- enforce\_2\_1 ( Quadrant ) - enforces the 2:1 balance condition
- partition ( Quadrant ) - partitions the quadrant until it meets the mesh control parameter of the topological elements in it.

The successive triangulation only needs the following functions:

- refine ( Quadrant ) - divides the perimeter of the quadrants into smaller segments

- `triangulate ( Quadrant )` - triangulates the quadrant and preserve the edges in it. This is a virtual function that needs to be overridden with a triangulation algorithm, such as Ruppert's Delaunay refinement algorithm, Paul Chew's Delaunay refinement algorithm, and a basic conforming Delaunay triangulation.

The mesh quality improvement only needs the following function:

- `mergeVertices( Quadrant )` - merge nearby vertices

## CHAPTER 3

### RESULTS

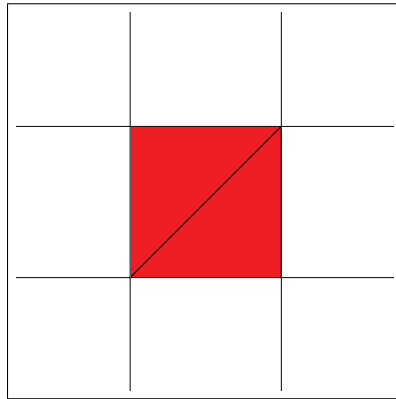
Before looking at the results, a systematic way to compare the quality of the triangulation is needed. This is needed to study the relationship between the mesh control parameter and the merge tolerance with the triangulation. Having a heuristic that quantitatively measures the quality of the triangles can achieve this. The heuristic needs to put the good quality elements on one end of the spectrum and bad quality elements on the other end of the spectrum. The quality assessment that is used in this research is:

$$Q(T) = \frac{4\sqrt{3} A(T)}{h_1^2 + h_2^2 + h_3^2} \quad (3.1)$$

where  $T$  is a triangle in the mesh,  $A(T)$  is the area of the triangle, and  $h_1$ ,  $h_2$ , and  $h_3$  denote the three lengths of the triangle's edges [8]. The range of this assessment yields from 0 to 1. The quality is at minimum for skinnier/collinear triangles, and the quality is at maximum for equilateral triangles. In addition to knowing the quality of the most desirable and undesirable elements, it is good to know the quality of some common triangles to this research. A common triangle to this research is the isosceles right triangle, or the 45°-45°-90° triangle. The quality of this triangle is 0.866. This triangle can be mostly found in quadrants that are interior or exterior to the domain. The other common triangles and their quality assessments are found in Figure 3.2. Figure 3.1 shows the quadrants that contain these common triangles.



### Interior/Exterior Quadrants



### Graded Quadrants

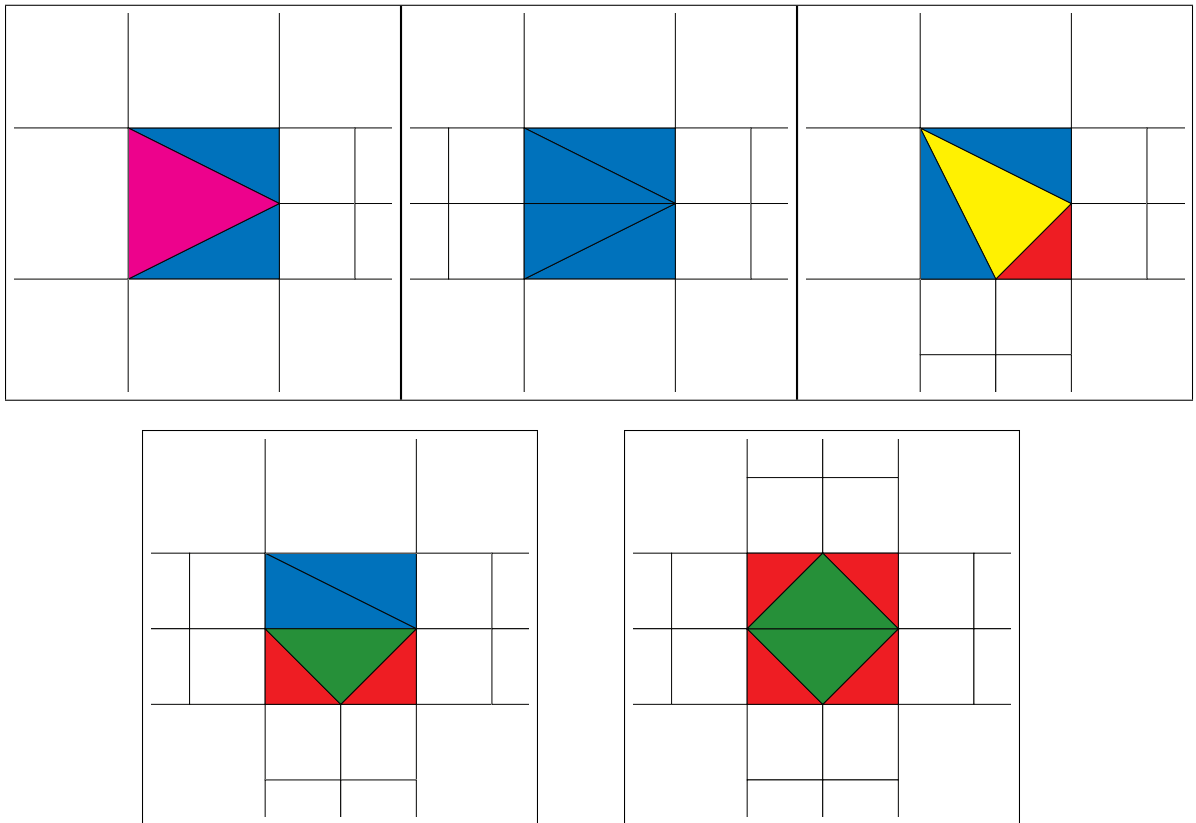


Figure 3.1: The compilation above composes all of the possible graded, interior, and exterior quadrants in a triangulation. These quadrants are very common in this work. Thus, the frequency of the common triangles is high because they can be found in these quadrants.

### Common Triangles

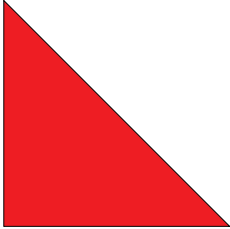
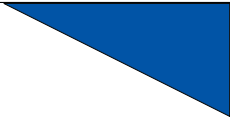
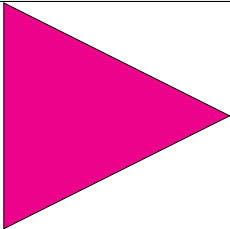
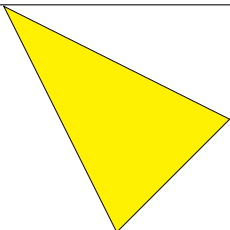
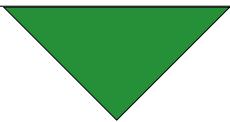
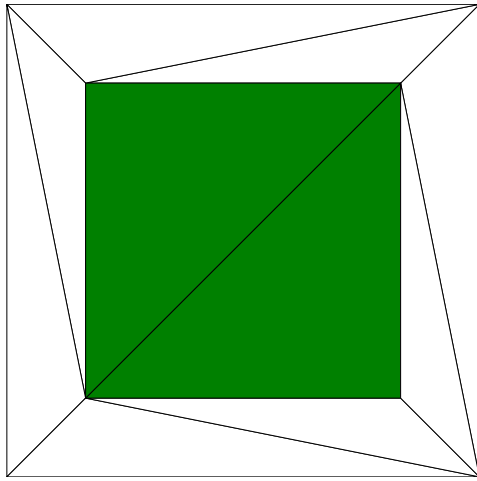
Picture	Name	Quality ( $Q$ )
	45°-45°-90° Triangle	$\frac{\sqrt{3}}{2} \approx 0.8660$
	1:2 Right Triangle	$\frac{2\sqrt{3}}{5} \approx 0.6928$
	Isosceles Triangle 01	$\frac{4\sqrt{3}}{7} \approx 0.9897$
	Isosceles Triangle 02	$\frac{\sqrt{3}}{2} \approx 0.8660$
	Isosceles Triangle 03	$\frac{\sqrt{3}}{2} \approx 0.8660$

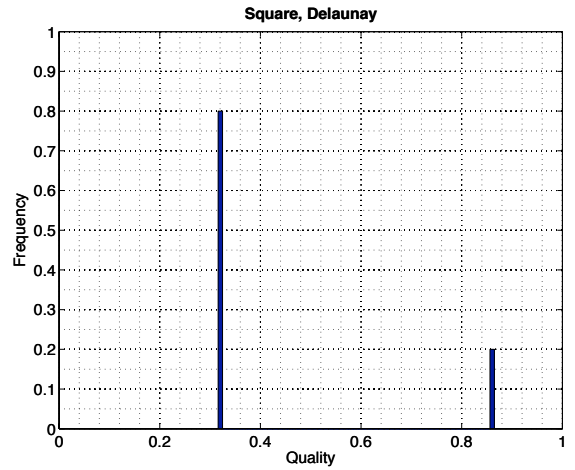
Figure 3.2: The list above calculates the quality  $Q$  of each common triangle.

### 3.1 Delaunay Triangulation

The conforming Delaunay triangulation algorithm was the first module to be implemented because this module is the easiest module to implement and has a straightforward functional requirement. Additionally, the functional requirement can be tested independently and thoroughly from the rest of the code. Figure 3.3 is a Delaunay triangulation of a square that is generated from this module, and Figure 3.4 is a conforming Delaunay triangulation of the serial Planar ultra-wideband modular antenna (PUMA) [33]. The red vertices in the triangulation are Steiner points that are inserted to preserve missing edge segments in the triangulation. No Steiner vertices are needed for the square because no edges are missing from the triangulation when all of the vertices in the model are inserted. In both of these triangulations, there are some poor quality elements, especially in the PUMA as shown in Figure 3.4(c).

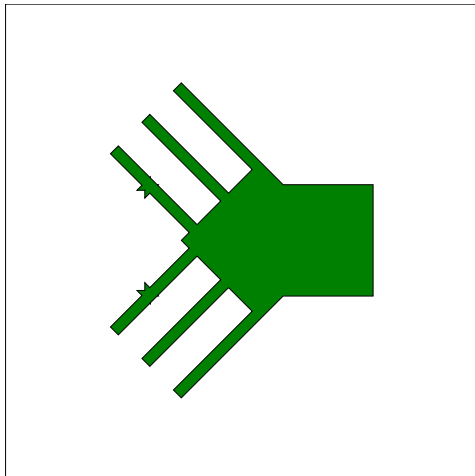


(a) Triangulation. Generated 10 triangles in 0.051 seconds.

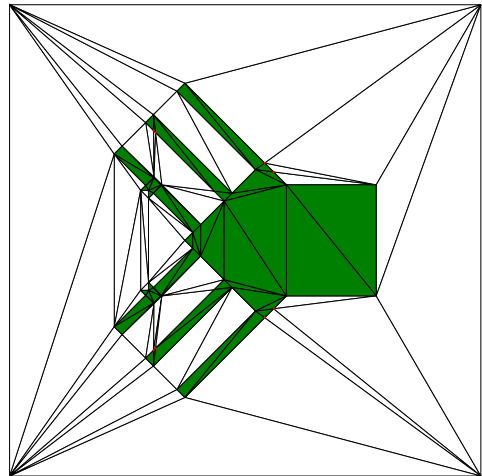


(b) Quality distribution.

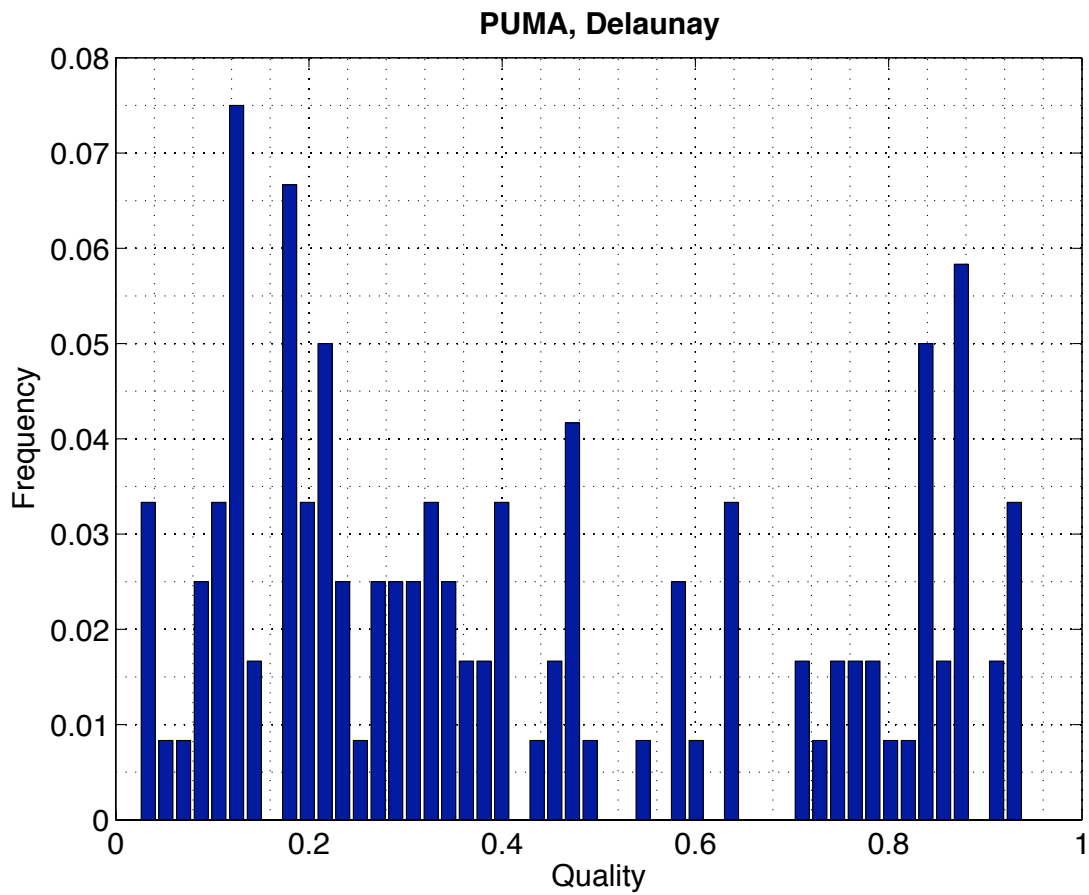
Figure 3.3: The conforming Delaunay triangulation of a square is shown on the left, and its quality distribution is shown on the right.



(a) Geometric model of PUMA.



(b) Triangulation. Red vertices are Steiner points that are inserted to preserve edges. Generated 120 triangles in 0.050 seconds.



(c) Quality distribution.

Figure 3.4: Conforming Delaunay triangulation of PUMA.

## 3.2 Uniform Grid

After the implementation of the successive triangulation process, an uniform subdivision of the entire computational domain is implemented as the spatial partitioning process. The uniform subdivision will subdivide all quadrants until its depths are equal to a common mesh control parameter. From these two implementations, the effects that mesh control parameter on the quality and approximation of the domain can be studied.

Figures 3.6 and 3.7 are triangulations of a continuous disk, shown in Figure 3.5, with the mesh control parameter 3 and 5 on the entire computational domain. The two figures demonstrate that increasing the mesh control parameter will produce better geometric approximation of the circular disk. This feature of adaptively conforming the mesh to parametric curves is a very attractive feature that has significant impact in FEM problems. Increasing the mesh control parameter will decrease the element size by definition, and it can also be seen in the two figures. In the respective quality distribution shown in Figures 3.6(b) and 3.7(b), increasing the mesh control parameter will increase the frequency of the triangles with the quality of approximately 0.866 in relative to the number of elements in its triangulation. These triangles with the quality of approximately 0.866 are mostly  $45^\circ$ - $45^\circ$ - $90^\circ$  triangles, which are mentioned earlier in this section. They are found mostly in quadrants that interior or exterior to the domain.

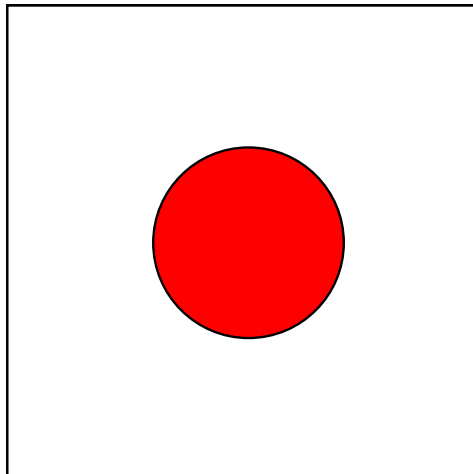
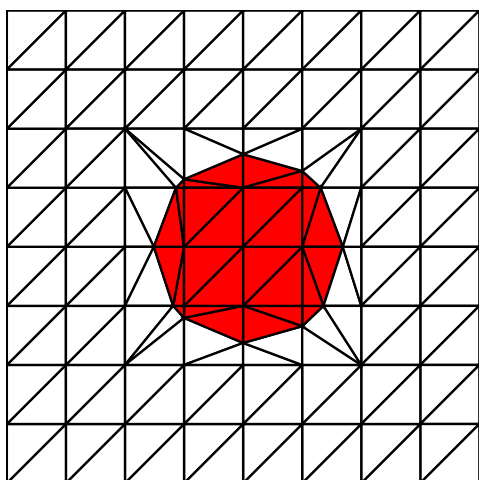
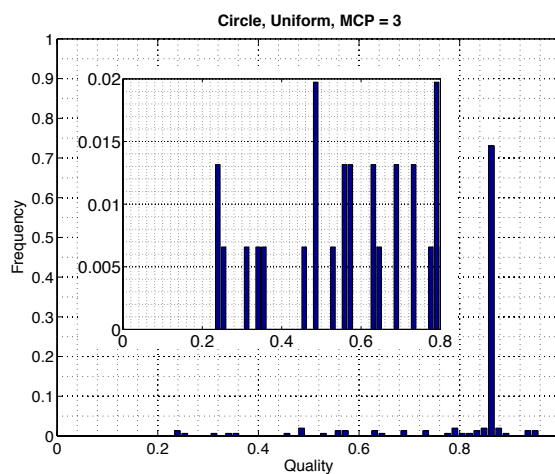


Figure 3.5: Smooth circular disk that is used to create the triangulations shown in Figures 3.6 and 3.7.

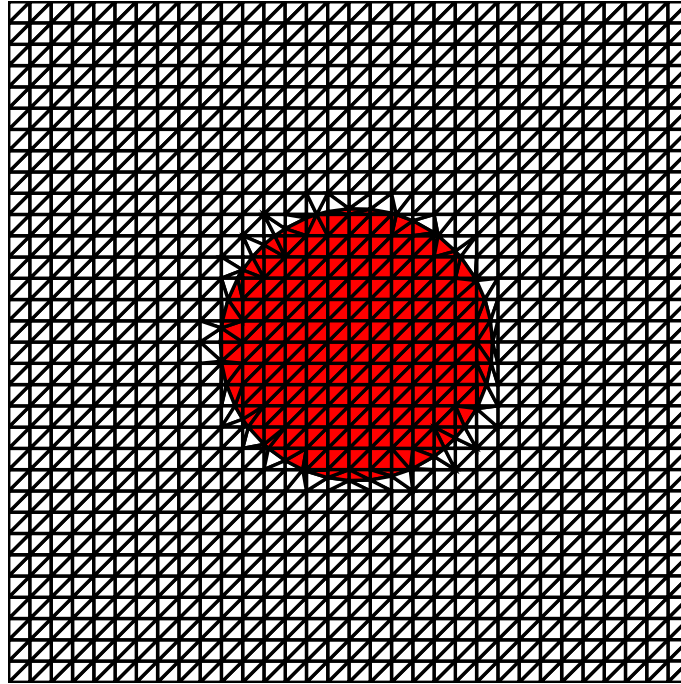


(a) Triangulation. Generated 152 triangles in 0.194 seconds.

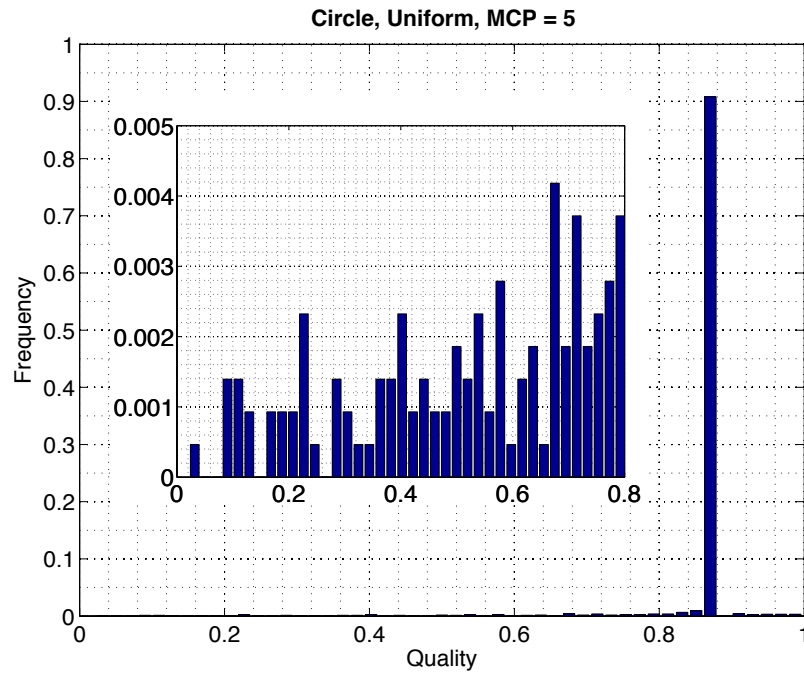


(b) Quality distribution.

Figure 3.6: Uniform quadtree triangulation of a circular disk inside a box as depicted in Figure 3.5. (a) Triangulation with uniform (balanced) quadtree spatial partitioning of level 3 (b) Mesh quality distribution.



(a) Triangulation. Generated 2152 triangles in 1.652 seconds.



(b) Quality distribution.

Figure 3.7: Uniform quadtree triangulation of a circular disk inside a box as depicted in Figure 3.5. (a) Triangulation with uniform (balanced) quadtree spatial partitioning of level 5 (b) Mesh quality distribution.

### 3.3 Non-uniform Refinement

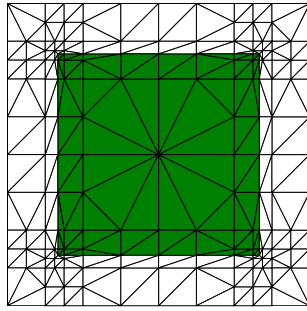
Figure 3.8(a) to (c) are the generated results from assigning a mesh control parameter on the vertices (vMCP), a mesh control parameter on the edges (eMCP), and a mesh control parameter on the faces (fMCP) respectively. Figure 3.8(d) is the histogram of their respective quality. Assigning the mesh control parameter on the face will put the minimal number of elements exterior to the domain. This option may be useful in FEM computations, where the field regularity can be known a-priori from the shape of the domain, and effectively leading to a naïve refinement strategy that could be used as the starting point for more advanced adaptive mesh refinement computations. Assigning the mesh control parameter on the edge will put the minimal number of elements everywhere except along the perimeter of the domain.

From Figure 3.8(c) to Figure 3.8(b), the triangulations do not look much different except for the interior of the model. The square in Figure 3.8(c) is uniformly triangulated with isosceles right triangles, and the square in Figure 3.8(b) is triangulated with graded elements. The quadrants that are interior to the square are empty. Therefore, each of these quadrants are triangulated with one of the graded quadrants that is shown in Figure 3.1. These quadrants will create only the triangles that are shown in Figure 3.2. There are three different qualities for these triangles: 0.866, 0.69, and 0.9897. As shown in Figure 3.8(d), this are the only three qualities where the frequencies of these two triangulations are different.

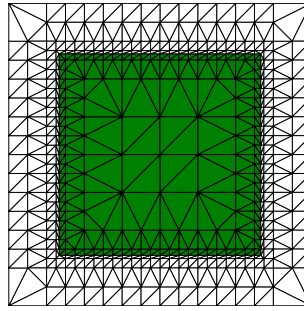
### 3.4 Merge Operations

Poor quality triangles may still result in denser meshed regions. Figure 3.9 highlights two poor quality triangles that contributed to the quality below 0.2 shown in Figure 3.7(b). However, point merges can eliminate these triangles and reduce the frequency at low qualities. Figure 3.10 highlights the same area as the Figure 3.9 and depicts that two poor quality triangles are eliminated by the point merge. The

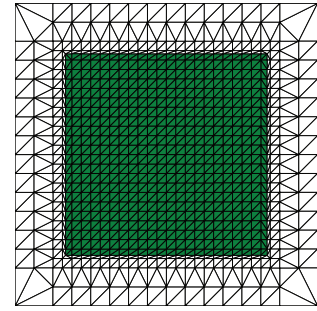




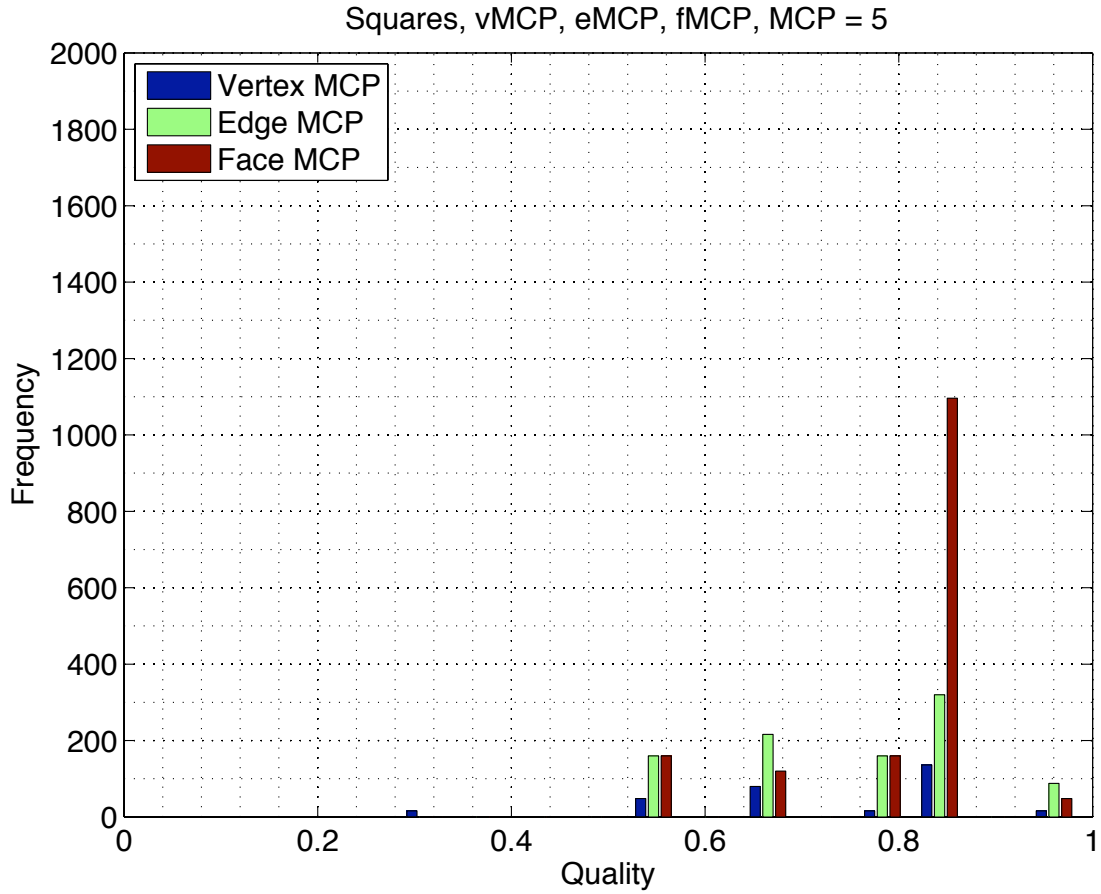
(a) Mesh control parameter is assigned on the vertices. Generated 312 triangles in 0.251 seconds.



(b) Mesh control parameter is assigned on the edges. Generated 944 triangles in 0.613 seconds.



(c) Mesh control parameter is assigned on the faces. Generated 1584 triangles in 1.100 seconds.



(d) Side-by-side comparison of quality

Figure 3.8: Non-uniform refinement. (a) Node refinement, where MCP is assigned to all four nodes. (b) Edge refinement, where MCP is assigned at the four edges of the square. and (c) Face refinement, where the MCP is assigned on the face of the square (d)Histogram of the resulting mesh qualities.

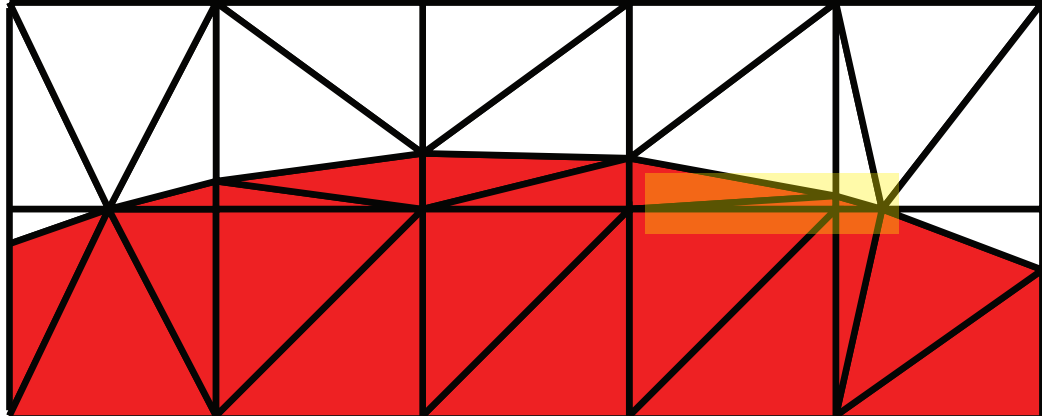


Figure 3.9: Poor quality triangles (highlighted in the yellow area) from the uniform triangulation of a circle in Figure 3.7 in a transparent yellow area.

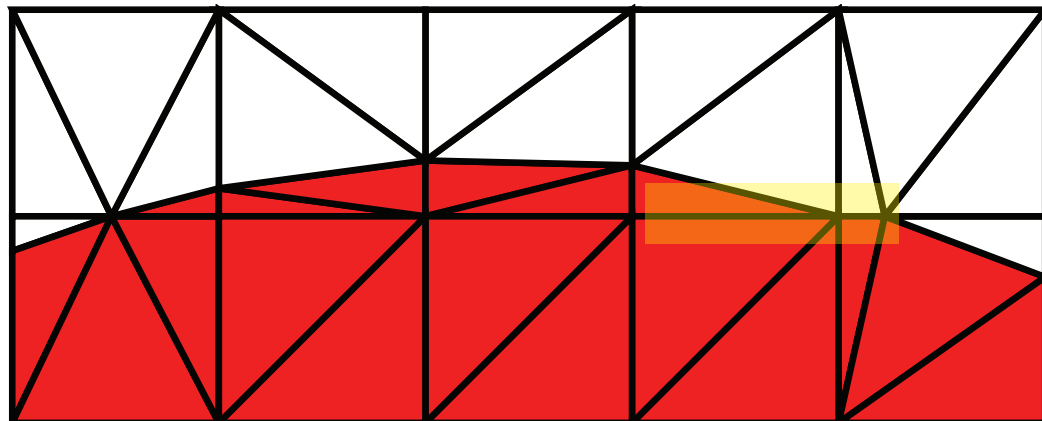
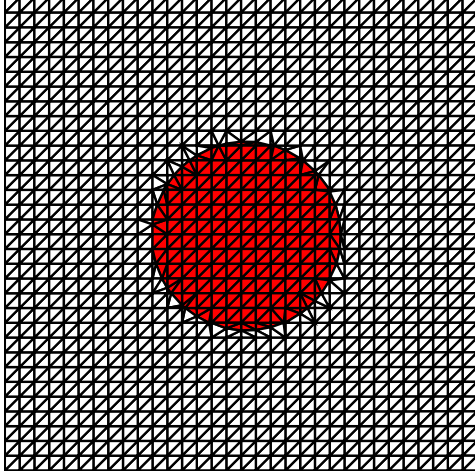
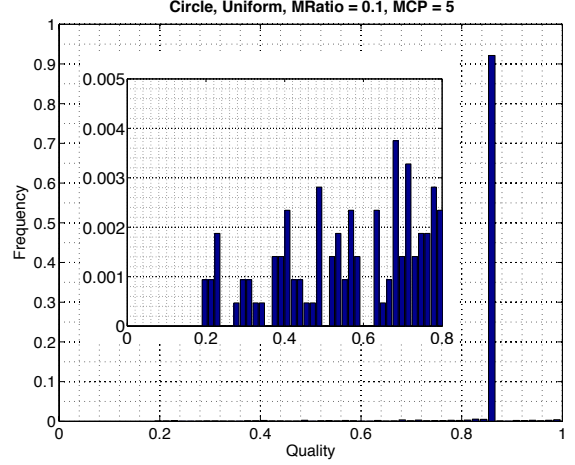


Figure 3.10: Improving the mesh quality of Figure 3.9 by merging geometry points to the quadtree corners and effectively eliminating the skinny triangles, but introducing some geometric error.

zoomed-in area shown in Figure 3.10 belongs to the triangulation shown in Figure 3.11. Its quality distribution is displayed on the right of the triangulation, and the merge tolerance for this example is 0.10. The quality distributions in Figure 3.11 and Figure 3.7 confirm that some poor quality elements are filtered out from merging nearby vertices. In this example, 18 triangles (of 2152 triangles in the triangulation without any point merges) were eliminated, and 11 (0.5%) of those triangles contributed to the 11 worst triangles in the triangulation.



(a) Triangulation. Generated 2134 triangles in 1.638 seconds.



(b) Quality distribution.

Figure 3.11: The same circle as in Figure 3.7 is uniformly triangulated with the same mesh control parameter, 5. In this case, the point merges are enabled and its merge tolerance is 0.10. The triangulation is on the left, and its quality distribution is on the right.

As illustrated in Figure 3.11(a), small deformations to the original approximation of the circular disk are formed due to the point merges. Visually, the original approximation (without merges) is closer to the original circle model than the approximation due to point merges. A comparison of their approximation errors can confirm this observation scientifically. The following error expression measures the approximation error for the circle:

$$\epsilon_{circ} = \frac{\int_0^{2\pi} \left| \frac{dA_m}{d\theta} - \frac{r^2}{2} \right| d\theta}{\pi r^2} \quad (3.2)$$

$\frac{r^2}{2}$  is the ideal  $\frac{dA}{d\theta}$  of the circle, and  $\frac{dA_m}{d\theta}$  is the measured  $\frac{dA}{d\theta}$  from the triangulation. Using this expression for the approximation error, the error for Figure 3.7(a) is 0.00363321, and the error for Figure 3.11(a) is 0.00464447. Figure 3.12 shows a plot (three sample points) of the relationship between the error and merge tolerance. The approximation error grows larger as the merge tolerance increases.

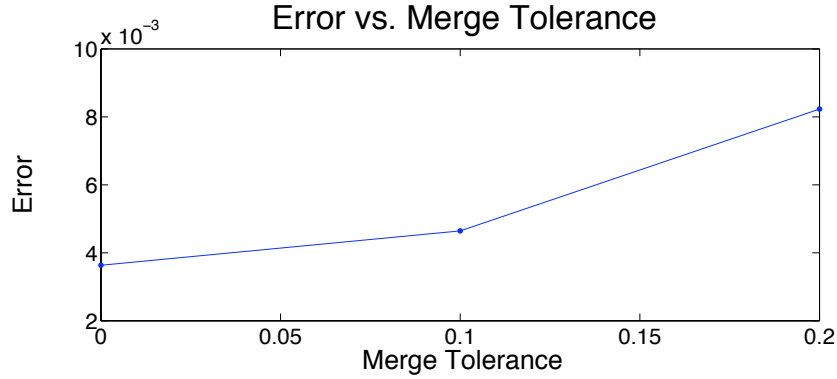
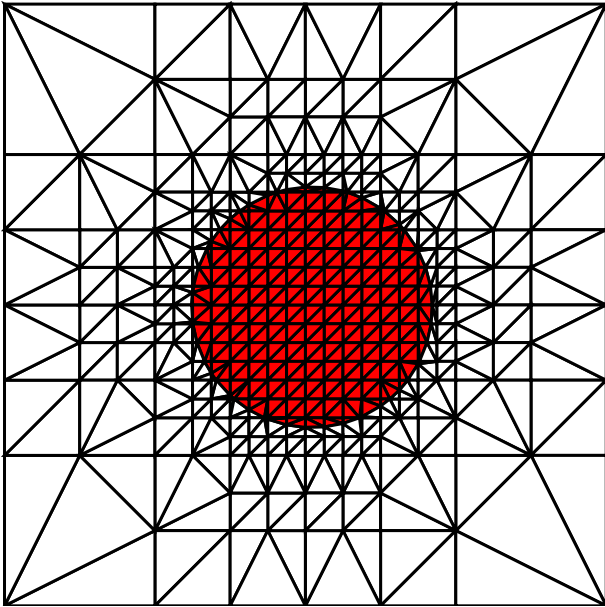
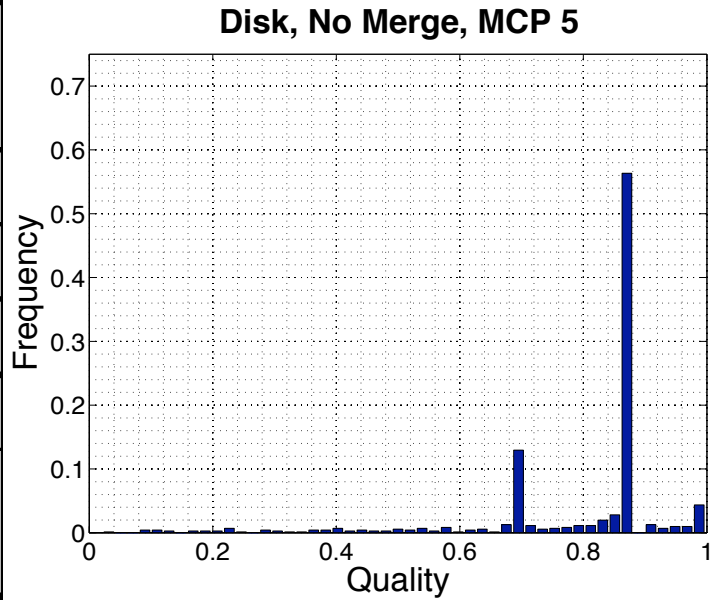


Figure 3.12: Plot of the relationship between approximation error and merge tolerance for the circle model.

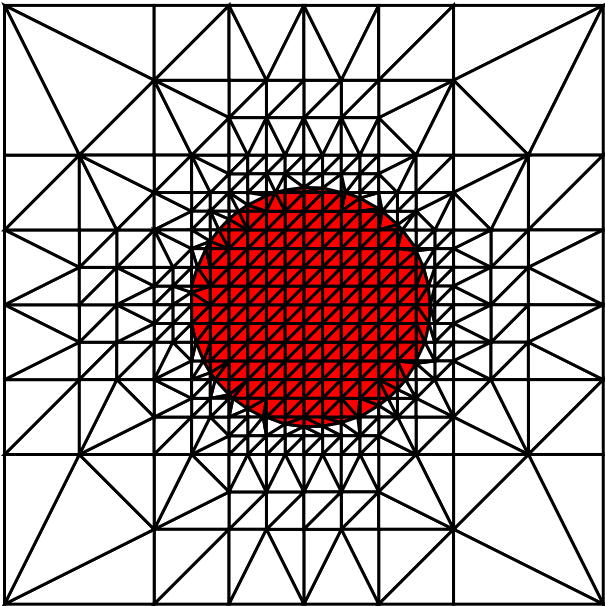
As previously discussed, increasing the merge tolerance filters out the poor quality elements by eliminating them. Figures 3.13(b), (d), and (f) shows that increasing the merge tolerance does more than reducing the frequency of poor quality elements; it completely filters out elements whose quality is lower than a cut-off threshold. The cut-offs are calculated using (2.4), and they are shown as a dotted red line in Figures 3.13(b), (d), and (f). These figures confirms that the cut-off is directly proportional to the merge tolerance because no elements below the cut-off exist. In the triangulations shown in Figure 3.13, there are no quadrants that have multiple intersections on a single edge or have more than two intersections. Additionally, there are no vertices from the domain in the circle model, and, thus, there are no vertices that are interior to a quadrant. Therefore, the calculated cut-off in (2.4) applies to the four triangulations in Figure 3.13. As shown in their histograms, there are no elements with a quality below the cut-off.



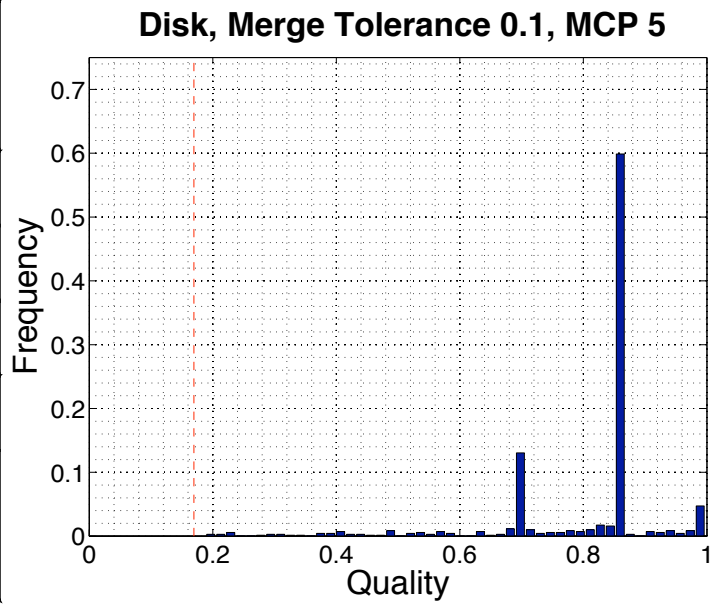
(a) No Merge. Generated 606 triangles in 0.452 seconds.



(b) Quality distribution.

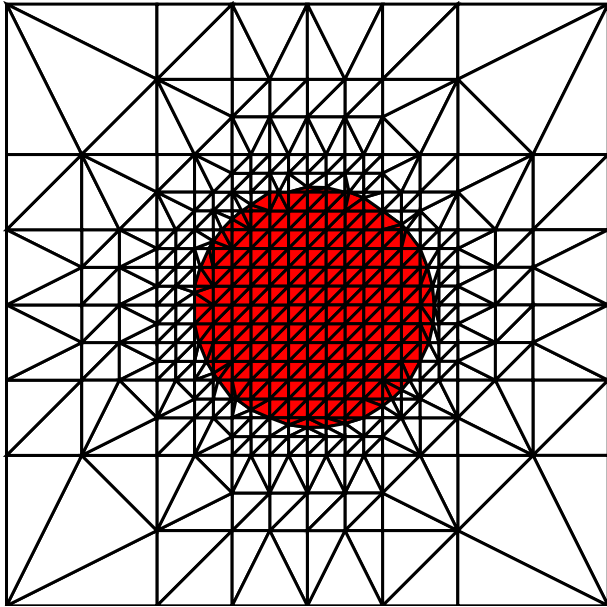


(c) Merge Tolerance = 0.10. Generated 604 triangles in 0.456 seconds.



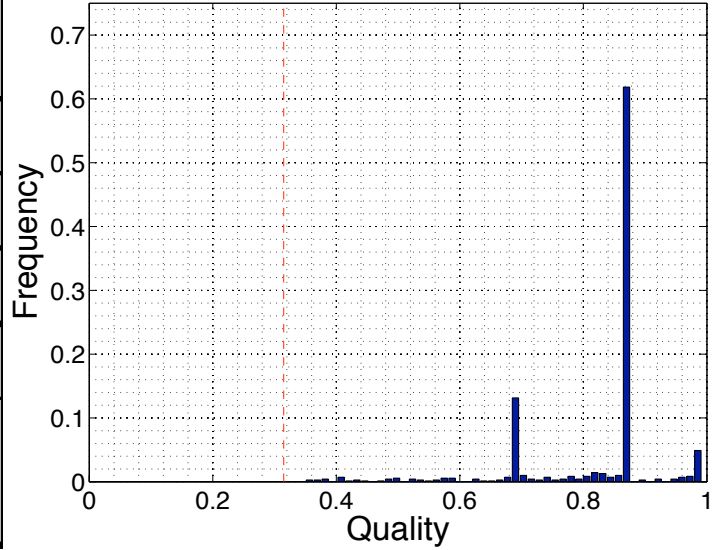
(d) Quality distribution.

Continue to next page for more results with higher merge tolerance.

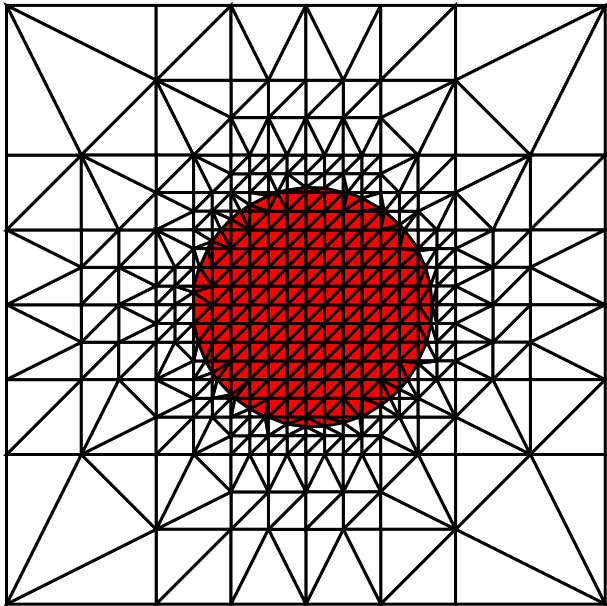


(a) Merge Tolerance = 0.20. Generated 601 triangles in 0.472 seconds.

**Disk, Merge Tolerance 0.2, MCP 5**

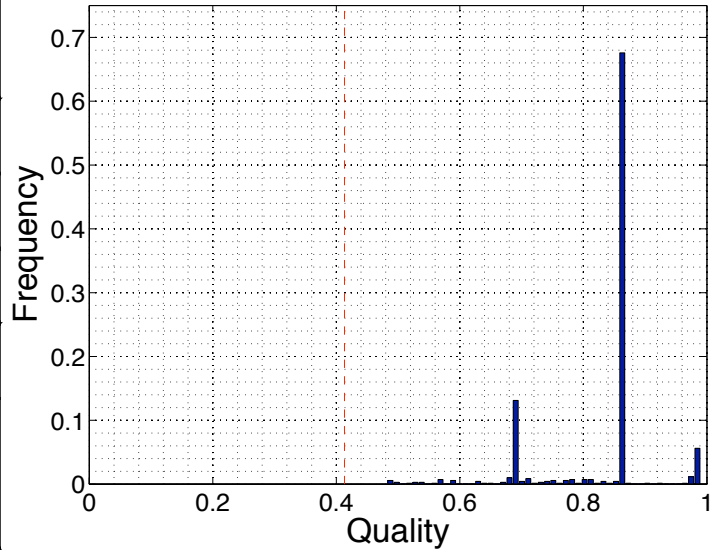


(b) Quality distribution.



(c) Merge Tolerance = 0.30. Generated 599 triangles in 0.491 seconds.

**Disk, Merge Tolerance 0.3, MCP 5**



(d) Quality distribution.

Figure 3.13: Effect of merge on mesh quality. (a) There are no calculated quality cut-off when the merge tolerance is 0. (b) The quality cut-off is at 0.1694. (c) The quality cut-off is at 0.3149. (d) The quality cut-off is at 0.4133.

### 3.5 Realistic Two-Dimensional Models

Figure 3.14 shows a model of a torus and its triangulation. The figure demonstrates that part of the triangulation can be removed. In this case, the hole in the torus was marked for removal. As shown in the triangulation, the hole is removed. There are no triangulations in the interior of the hole. An user may be interested in this option if the user are not interested in the elements in a particular area. Figure 3.15 shows a triangulation of a printed circuit board (PCB) [34]. The mounting holes on the PCB are removed in its triangulation because they are of no interest.

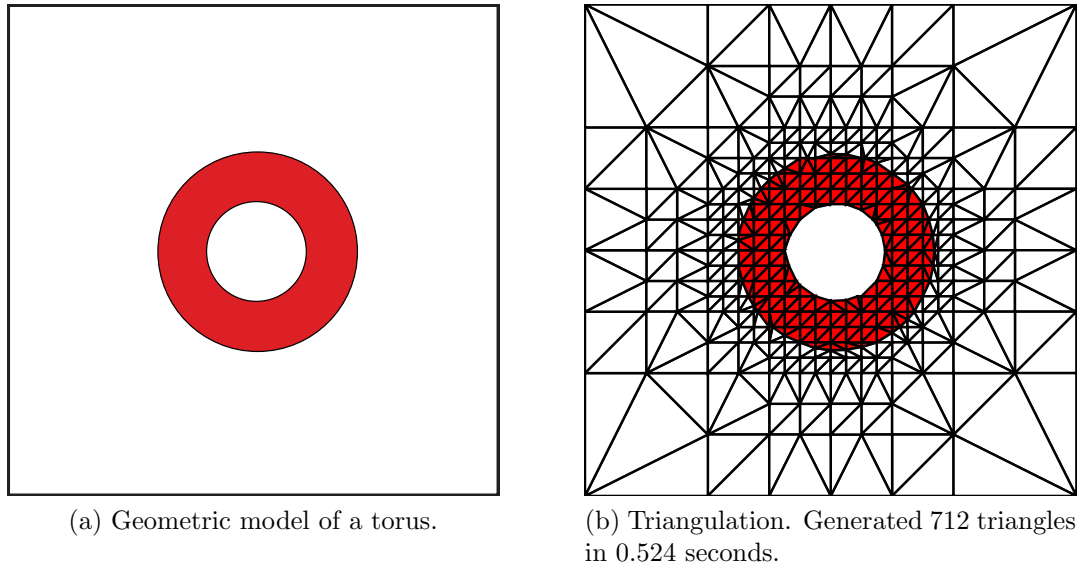
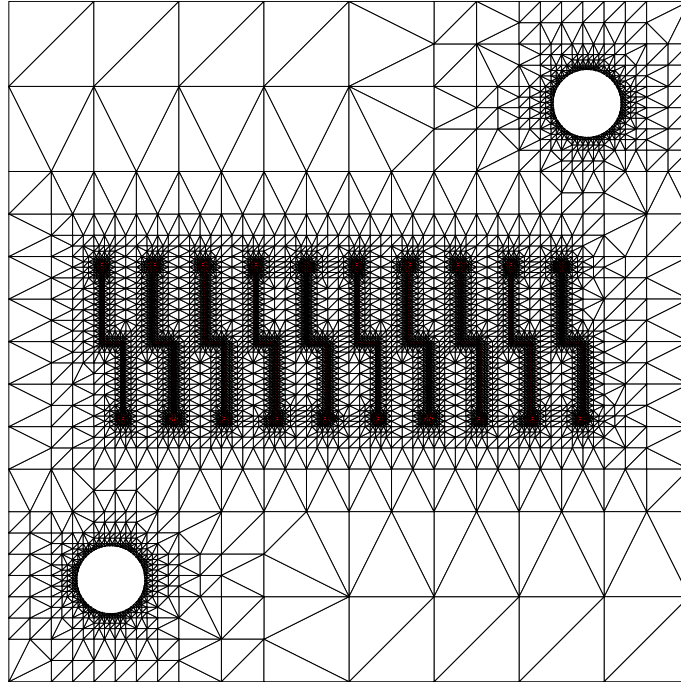


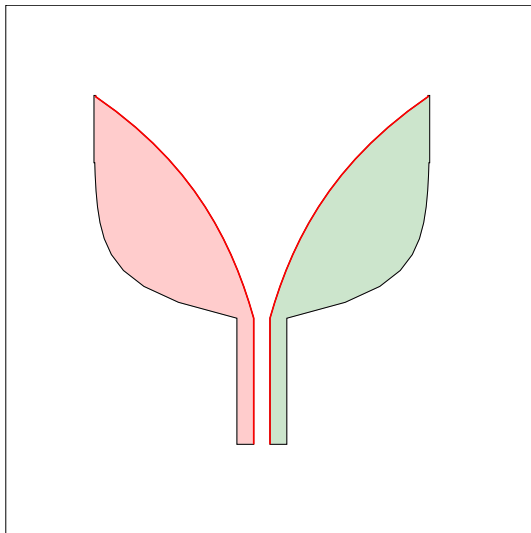
Figure 3.14: Triangulation of a torus. The mesh generator can mesh shapes with holes and remove them from the triangulation. (a) Geometric model of a torus with a  $fMCP = 5$ . (b) Triangulation of a torus.

All of the demonstrated examples up until now have the same mesh control parameter assigned to all of its edges. Figure 3.16 shows it is possible to assign mesh control parameters to a subset of edges in the domain. The red edges in Figure 3.16(a) are the only edges that are assigned mesh control parameters. In this example, a mesh control parameter of 7 is assigned to all of the red edges, and its triangulation is shown in Figure 3.16(b).

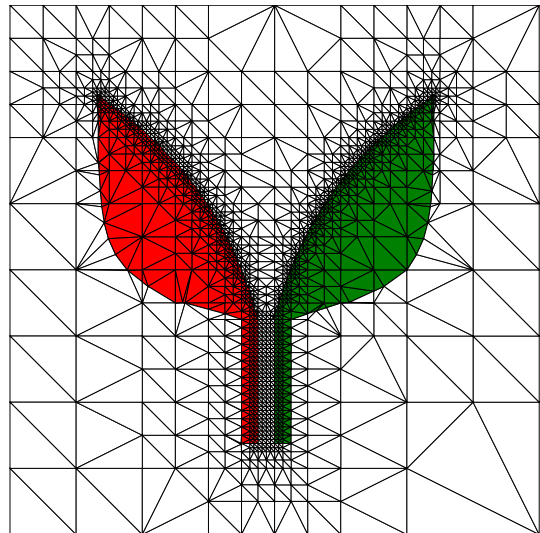


(a) Triangulation with merge tolerance = 0.1. Generated 16620 triangles in 11.137 seconds.

Figure 3.15: Triangulation of traces on printed circuit board (PCB). The mounting holes are removed from the triangulation.



(a) The red edges are the only entities assigned mesh control parameter. The red edges have the mesh control parameter 7.



(b) Triangulation of BAVA. Generated 3298 triangles in 2.206 seconds.

Figure 3.16: Different mesh control parameters on different topological elements. (a) Model of balanced antipodal Vivaldi antenna (BAVA) [35] (b) Triangulation of BAVA.



Figure 3.17 is a model of the exponential flare piece from a Vivaldi antenna [36]. Figure 3.18 shows a triangulation of the Vivaldi flare with the merge tolerance of 0.2. On the right figure, the generated mesh is not topologically consistent with the original Vivaldi flare because there are two shapes in the output whereas there is one in the model. The two shapes resulted in the output because the sampled points at the thin area are merged to the same points. Increasing the mesh control parameter on the two edges will recover the thin area. This is shown on Figure 3.18(a), whose mesh control parameters on the edges increased from 4 to 6.

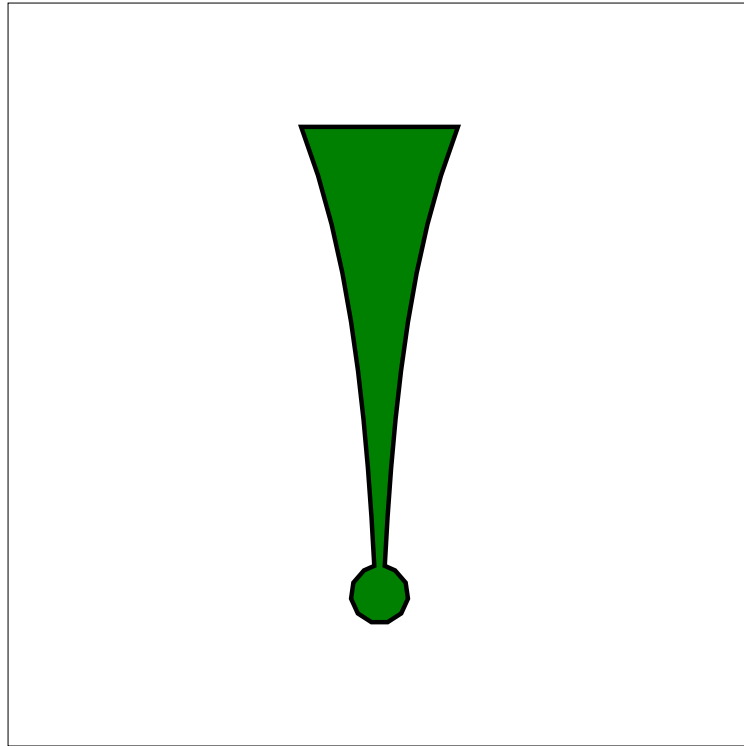
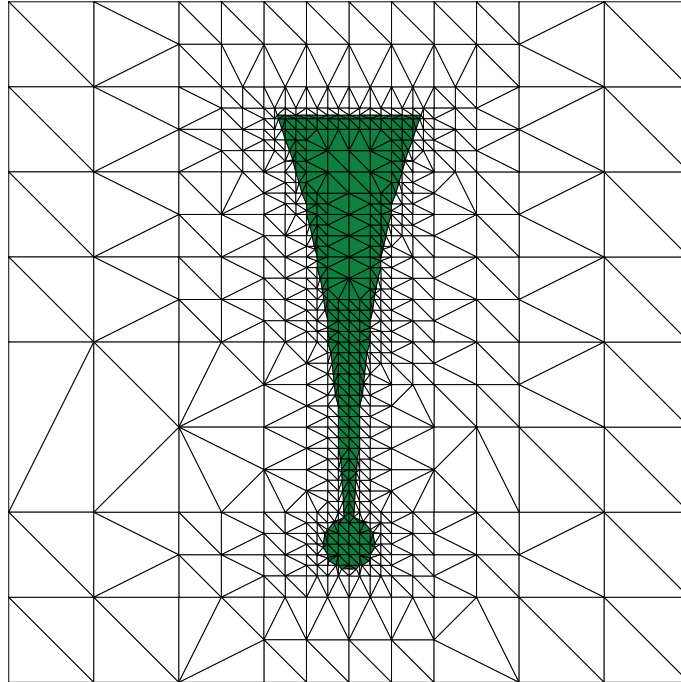
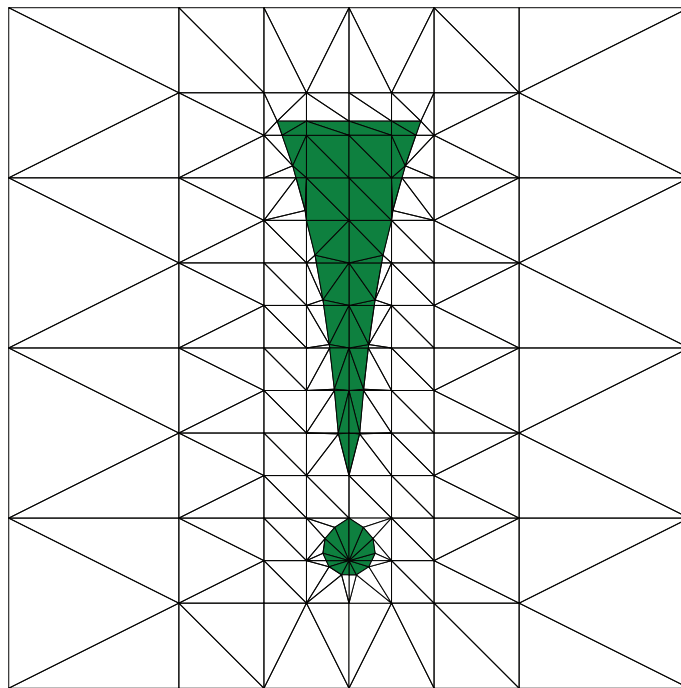


Figure 3.17: Model of the flare piece from a Vivaldi antenna.



(a) Mesh control parameter 6 is assigned to all edges of the domain. Generated 1319 triangles in 1.07 seconds.



(b) Mesh control parameter 4 is assigned to all edges of the domain. Generated 268 triangles in 0.261 seconds.

Figure 3.18: Topological inconsistencies resulting from poor usage of MCP and merge tolerance. (a) Case where the merge=0.2 and the eMCP=6, leading to a topologically consistent mesh. (b) Case where the merge=0.2 and the eMCP=4, leading to a topologically inconsistent mesh.

### 3.6 Discretized Curve and Smooth Curves

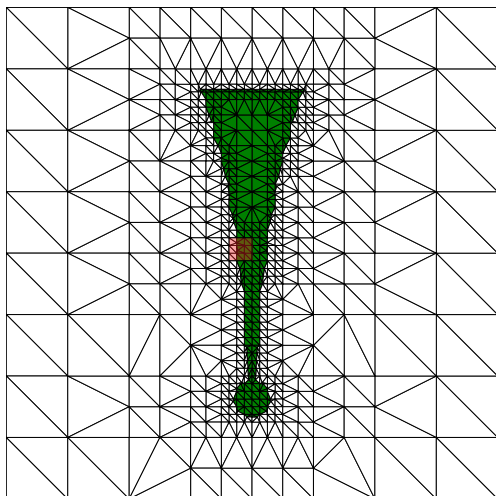
A smooth curve may have many advantages over its discretized counterpart. A discretized curve at input may lead to bad quality elements because there are no lower bound on the quality on quadrants that contain a vertex from the domain with point merges as discussed in the previous chapter. Figure 3.18(b) shows a similar scenario as Figure 2.20 in a practical run. Such problems do not exist in the case of a smooth curve. To show this, Figure 3.18(e) compares the quality distribution of the two versions side-by-side. The discrete Vivaldi flare has four bars below 0.4 whereas there is none for the smooth Vivaldi flare. Other than the four bars below 0.4, the shape of the two distribution are identical.

The number of discretized edges and vertices increases the modeling accuracy and the approximation of its original model. The approximations are fixed for all mesh control parameters in the triangulation without point merges. Table 3.1 shows that the errors are approximately equal across all mesh control parameters for a fixed number of discretized points. However, the merging process moves vertices around to eliminate bad quality elements. In almost all cases, the approximation will be worse than it is without the merging process. Increasing the mesh control parameter will restore the approximation because some segments of the edges in the design model are restored.

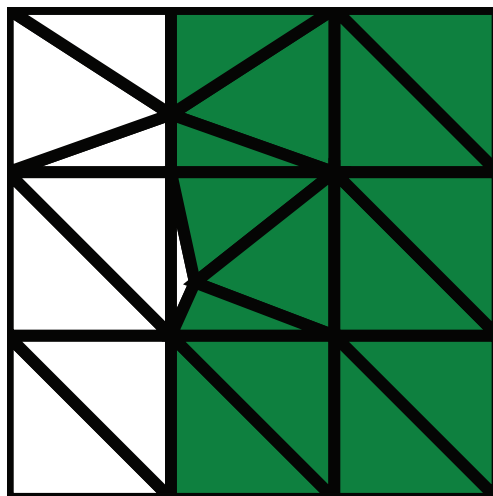
**Approximation Error Due to Number of  
Discretized Points and Mesh Control Parameter**

Number of Discretized Points	MCP				
	2	3	4	5	6
10	0.0630883	0.0608727	0.0624917	0.0645107	0.0641969
20	0.0156848	0.0152223	0.014714	0.0161502	0.0163587

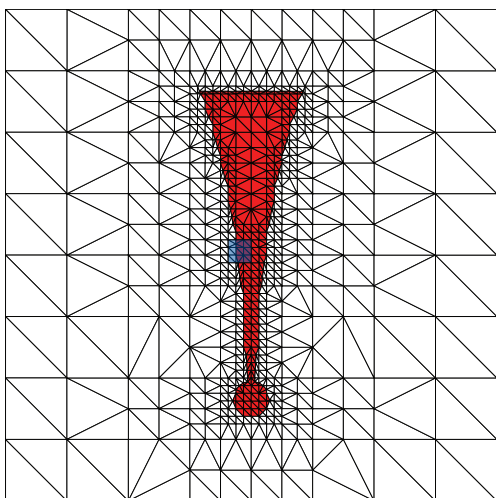
Table 3.1: Approximation error due to the number of discretized points and mesh control parameter.



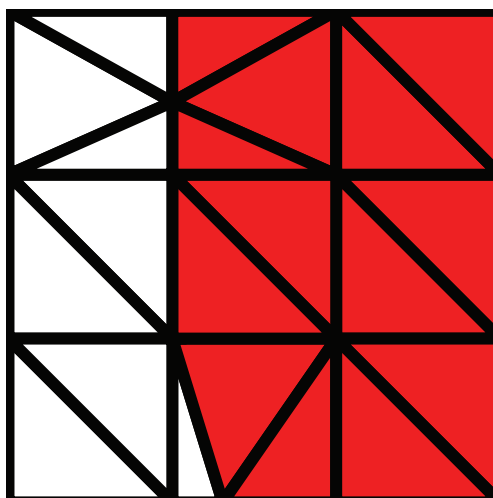
(a) Discrete curve. Generated 1310 triangles in 1.204 seconds.



(b) Zoomed area of the highlighted area in the left figure.

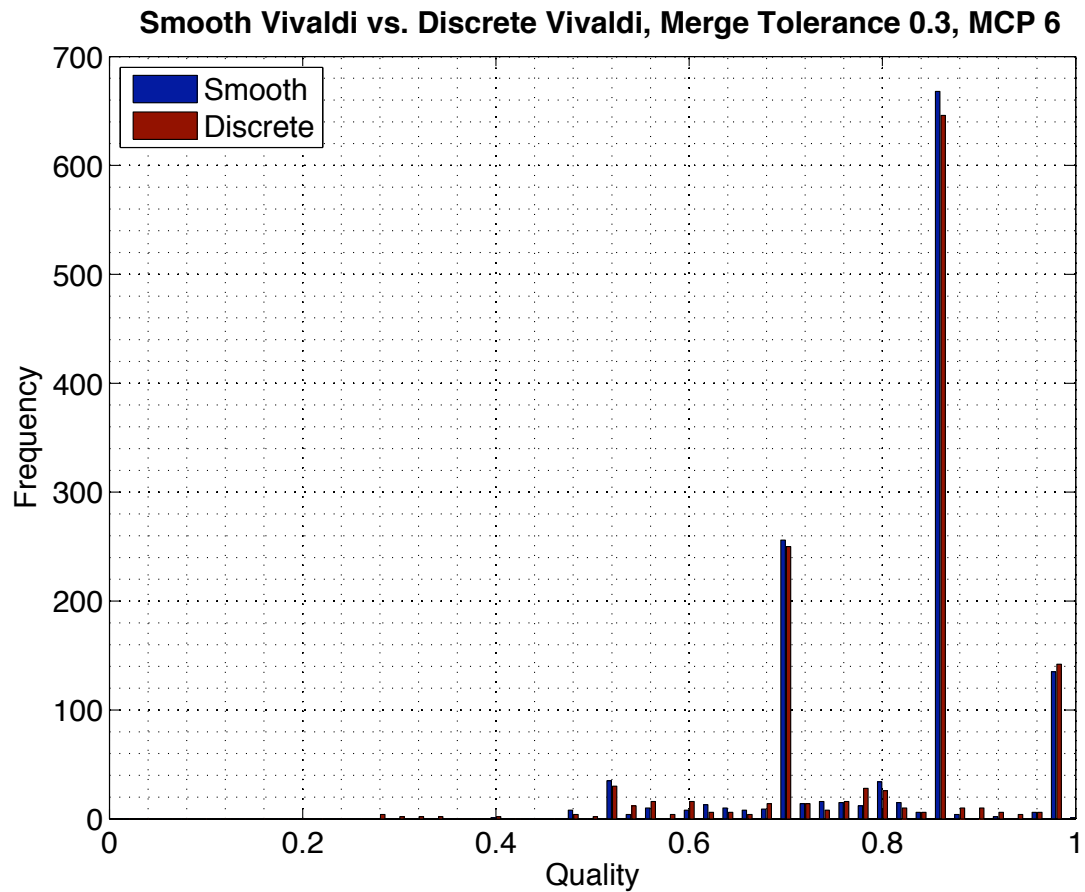


(c) Smooth curve. Generated 1290 triangles in 1.101 seconds.



(d) Zoomed area of the highlighted area in the left figure.

Continue to next page for the quality distribution.



(e) Side-by-side comparison of the smooth Vivaldi flare and the discretized Vivaldi flare.

Figure 3.18: Meshing of discretized vs. smooth parametric curves. In all cases, the merge tolerance is 0.3 and the edge MCP=6. (a) Mesh of discrete parametric curve. (b) Zoomed view of a region around the curve (a). (c) Mesh of the smooth parametric curve (d) Zoomed view of a region around the curve in (c). (e) Side-by-side comparison of the mesh quality histograms, showing the smooth parametric curve model has the better mesh quality.

Using smooth curves may not necessarily produce a better approximation of the domain than discretized curves at a low mesh control parameter. At a greater mesh control parameter, mesh created from smooth curves, if exactly model the ideal curves of the domain, will always approximate the domain better than discretized curves. Unlike discretized curves, increasing the mesh control parameter will increase its approximation whether there are point merges.

Figure 3.19 is a model of the Vivaldi with both its conducting piece and dielectric piece, and its triangulations are shown in Figure 3.20. Figure 3.20(a) is a triangulation of the Vivaldi that has the same merge tolerance in the entire domain. The left and right side of the flare are the only topological elements that carry a non-zero mesh control parameter. Segments of the left and right edges conducting piece (red face) merge to the grid because segments of both edges are in quadrants of different sizes. Users may find this undesirable. To fix this, each topological element has its own merge tolerance. In Figure 3.20(b), the left and right edges of the conducting piece have the merge tolerance of 0.

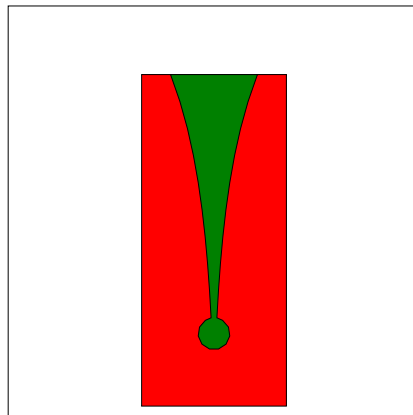


Figure 3.19: Model of the Vivaldi antenna, with both conducting (red) and dielectric (green) pieces.

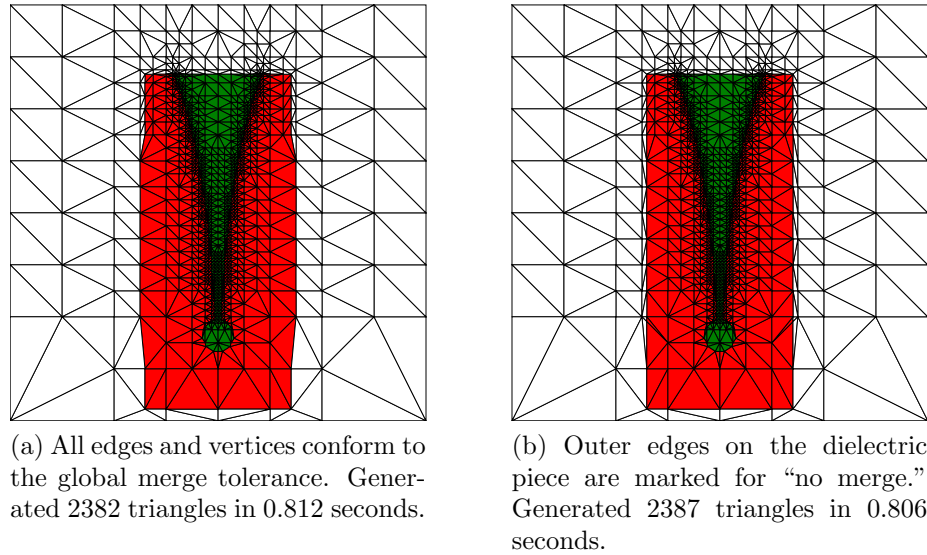


Figure 3.20: User enabled merge option for a Vivaldi antenna. (a) Merging all edges of the model to the quadtree nodes. (b) selectively merging only the exponential flare and circular region edges.

### 3.7 Handling Models from a Poor Quality CAD Tool

When creating a model on a poor quality CAD tool, it is possible for a created model to be topologically different from the model that the user is trying to make. For example, in Figure 3.21, the user is trying to create two touching squares. Instead, the user created two shapes that are not geometrically squares. Figure 3.21(a) zooms in on the spacing of the two points that should have been the same vertex. The two points are far enough apart to be considered as two different topological vertices. Both of the conforming Delaunay and Delaunay refinement algorithms have their own troubles meshing this problem. The conforming Delaunay triangulation has a very skinny triangle in its triangulation because the worst quality is in the thousandths digit. The Delaunay refinement algorithm created many triangles in an attempt to improve the quality of this skinny triangle. Using this combined quadtree and Delaunay method, as shown in Figure 3.21(d) and Figure 3.21(e), this problem is treated as two touching shapes due to the point merges. Under the corresponding figures, the measured worst quality and the number of elements are listed. With this

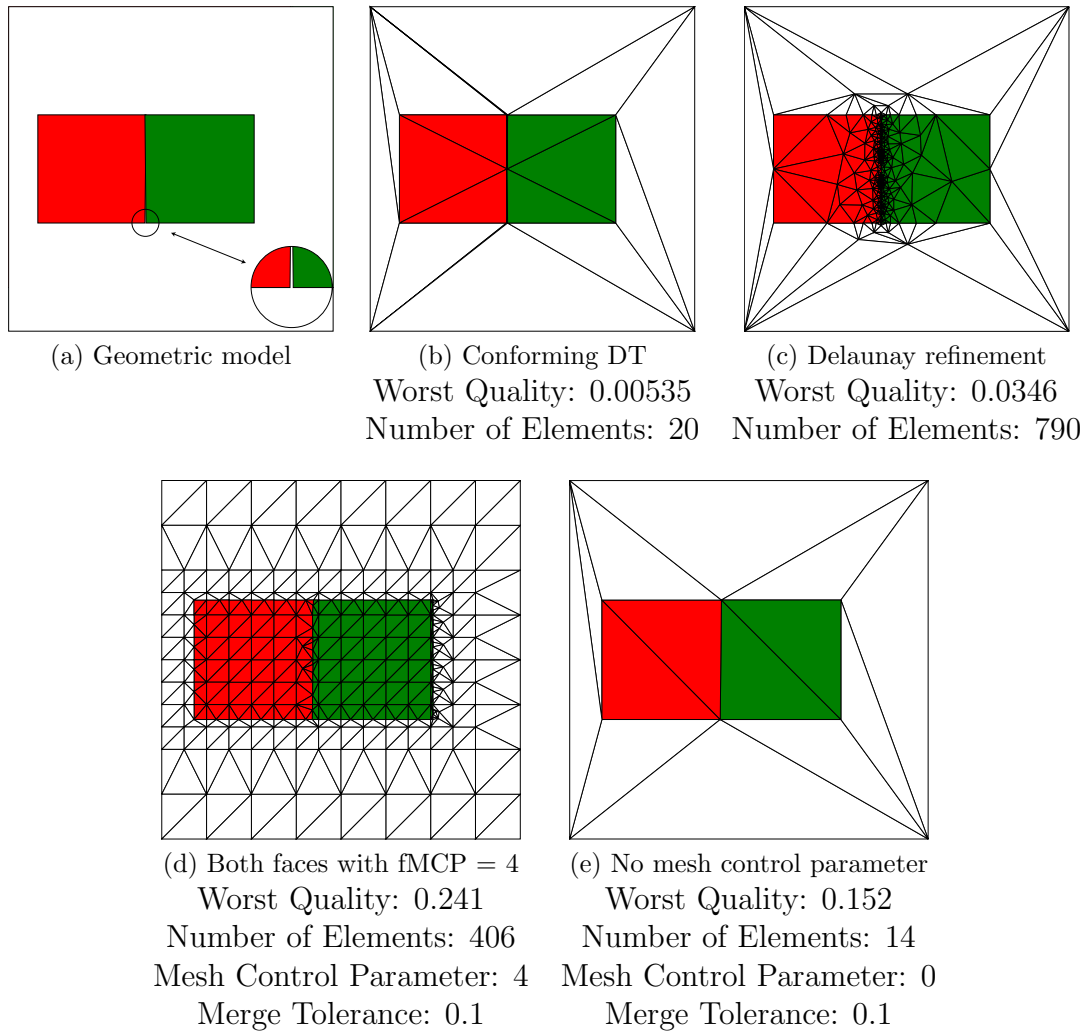


Figure 3.21: Model created from a poor quality CAD. (a) Two touching squares are created in a poor quality CAD tool. The circle zooms in on the bottom region of the “supposedly” touching edges. (b) Conforming Delaunay triangulation of the model shown in Figure (a). (c) Delaunay refinement triangulation of the model shown in Figure (a). (d) The two squares are touching in the triangulation because the vertices along the touching edges merges to a vertex on the other square. The two squares have  $fMCP = 4$ . (e) The two squares are touching in the triangulation because the vertices along the touching edges merges to a vertex on the other square. The two squares have no mesh control parameters.



combined method, the smallest quality increases, and the user have control over the number of elements by controlling the mesh control parameter.

### 3.8 Elements vs. CPU Time

The graph in Figure 3.23 plots the CPU time to generate a triangulation of the serial Planar ultra-wideband modular antenna (PUMA) [33] as a function of the number of elements. Figure 3.22 illustrates the model of the PUMA and its triangulation. Both spatial partitioning process and successive triangulation (with and without merge) have the time complexity of  $O(n \log n)$ . The graph shows the run time of the successive triangulation process is greater than the run time of spatial partitioning process at greater mesh size. As mentioned in the previous section, the triangulation of the quadrant is independent of the triangulation of another quadrant. As a result, the success triangulation can be executed in parallel. However, the successive triangulation is done serially here, and the the execution time will be greatly reduced if it had been implemented to run in parallel. At smaller mesh size, the successive triangulation finishes more quickly than the spatial partitioning process. Table 3.2 shows the recorded numerical time of the run at various mesh control parameters. At mesh control parameters below 8, the recorded times of the successive triangulation is less than the recorded time of the spatial partitioning process. This combined method will be compared against Shewchuk’s mesh generator “Triangle” [16] because “Triangle” is one of the most well known two-dimensional mesh generator. Therefore, in comparison to ”Triangle”, this mesh generator takes approximately 10 times longer to generate a mesh.

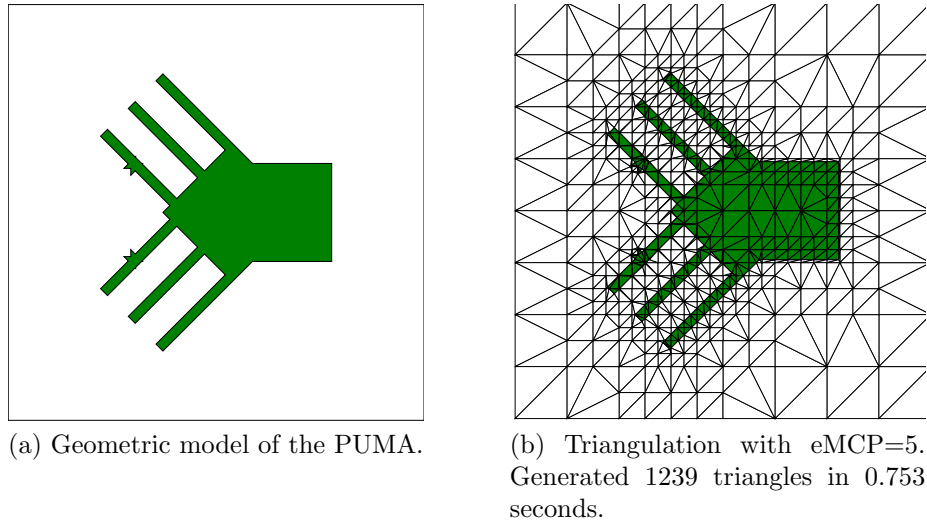


Figure 3.22: Model and triangulation of the PUMA.

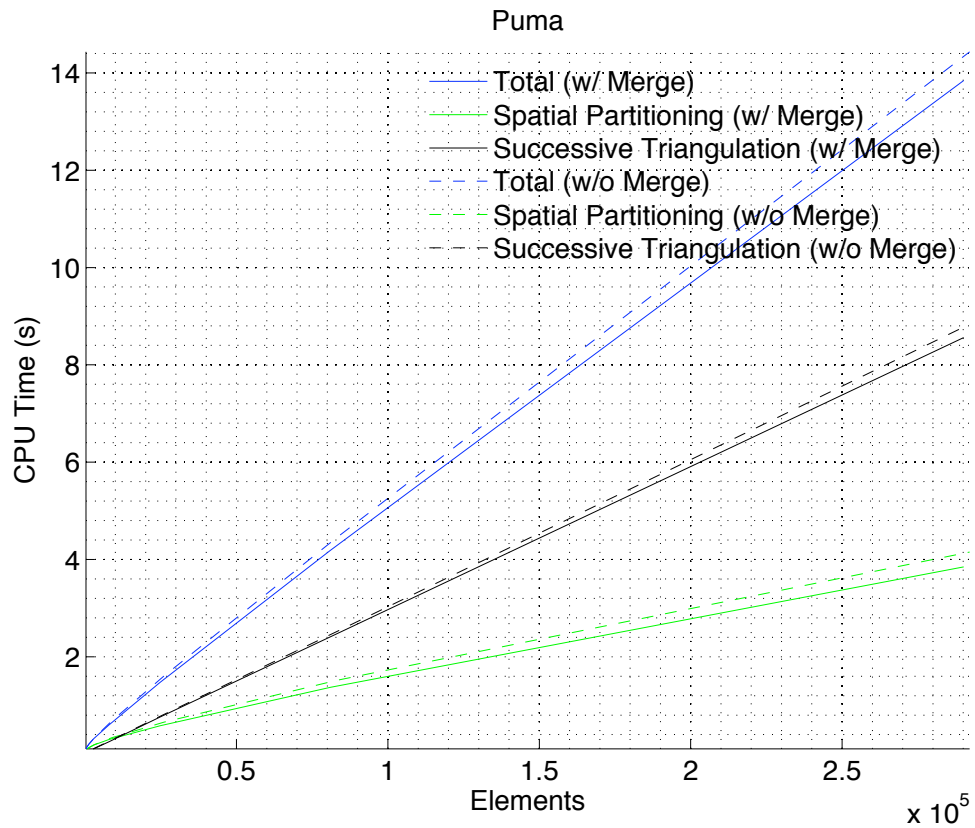


Figure 3.23: CPU time vs. number of generated elements for the PUMA example. The figure also compares the cases with and without merge. Both curves for the spatial partitioning and triangulation are  $O(n \log n)$ .

MCP	CPU Time (seconds)					
	No Point Merges			Merge Tolerance = 0.30		
	Partition	Triangle	Total	Partition	Triangle	Total
3	0.067513	0.031052	0.107759	0.072432	0.027553	0.108264
4	0.090464	0.039371	0.140695	0.095089	0.032293	0.136708
5	0.128763	0.061722	0.204646	0.129539	0.054609	0.19737
6	0.193367	0.120157	0.336218	0.185057	0.105719	0.312138
7	0.334184	0.290727	0.675689	0.297899	0.247918	0.591437
8	0.656288	0.814487	1.60603	0.58241	0.759361	1.47167
9	1.51417	2.51505	4.44364	1.3718	2.40902	4.18529
10	4.15294	8.83598	14.4356	3.84706	8.56187	13.8425

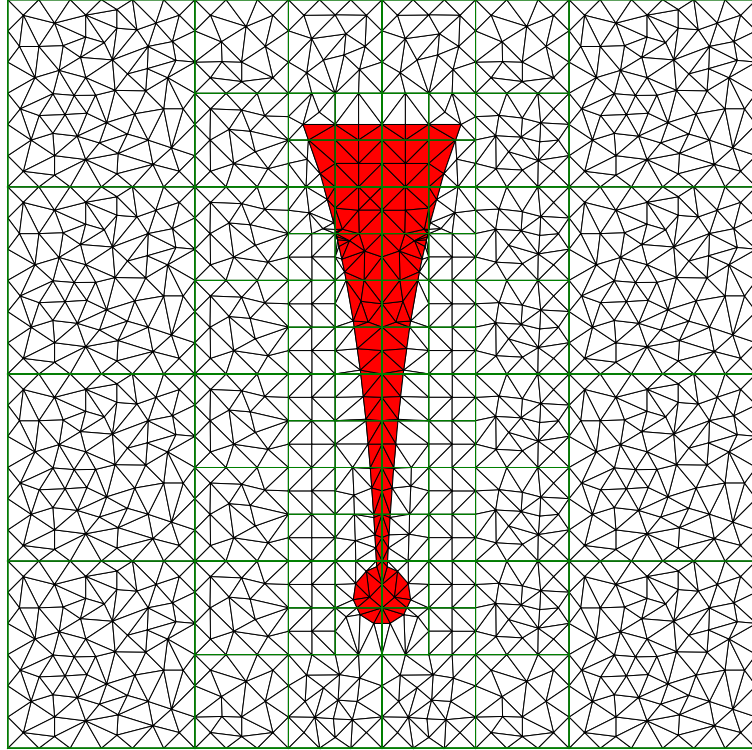
Table 3.2: Table shows the recorded CPU times for different mesh control parameter.

### 3.9 Delaunay Refinement as Successive Triangulation Method

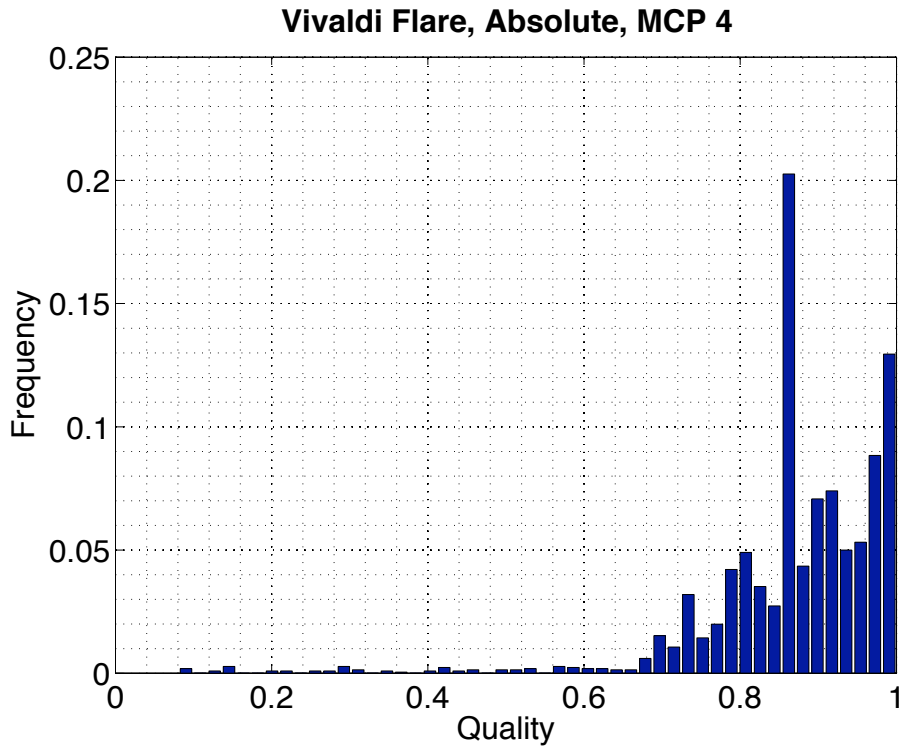
Finally, users have the option to substitute the conforming Delaunay triangulation algorithm with a Delaunay refinement algorithm, and, thus, using the mesher as a mesher/mesh partitioning tool appropriate for domain decomposition FEM computations. Figure 3.24 shows a mesh of a Vivaldi flare that is generated by the combined quadtree and Delaunay method using a Delaunay refinement algorithm in the successive triangulation module instead of a conforming Delaunay triangulation algorithm. As mentioned in the introduction, a Delaunay refinement algorithm produces a conforming Delaunay triangulation. In addition to inserting Steiner points to preserve edges, Delaunay refinement algorithms also insert vertices into the triangulation to improve the quality in its triangulation.

Specifically, Shewchuk's "Triangle" [16] was used as the successive triangulation algorithm in the implementation to generate Figure 3.24. Users have the option to control the size of the elements produced by Triangle. In Figure 3.24, the maximum area of the elements is 1 square unit, and the size of the bounding box is 113.74-by-113.74 square units. In Figure 3.25, the maximum area of the elements is relative to the area of the element's quadrant. The relative ratio between the maximum element area and the area of the quadrant in Figure 3.25 is 0.1. In both of these plots, the

quality distribution skewed more to the right than the quality distribution of a similar triangulation in Figure 3.1. For the fixed maximum element area, the refinement will only affect the quality of the triangles in the larger quadrants. The Delaunay refinement algorithm will become a conforming Delaunay triangulation algorithm for smaller quadrants. Increasing the mesh control parameter will cause the distribution to look like the distributions that are triangulated with conforming Delaunay triangulation algorithm as the successive triangulation. In Figure 3.24(b), the peak at 0.866 is becoming apparent. For the relative maximum element area, the distribution is skewed toward the right, and the peak at the two of the three common qualities, 0.693 and 0.866, are not so apparent. The quality 0.999 is the peak of Figure 3.25(b), but it is not due to the gradient quadrants, which create these elements in the previous examples. These elements are created by the Delaunay refinement algorithm.

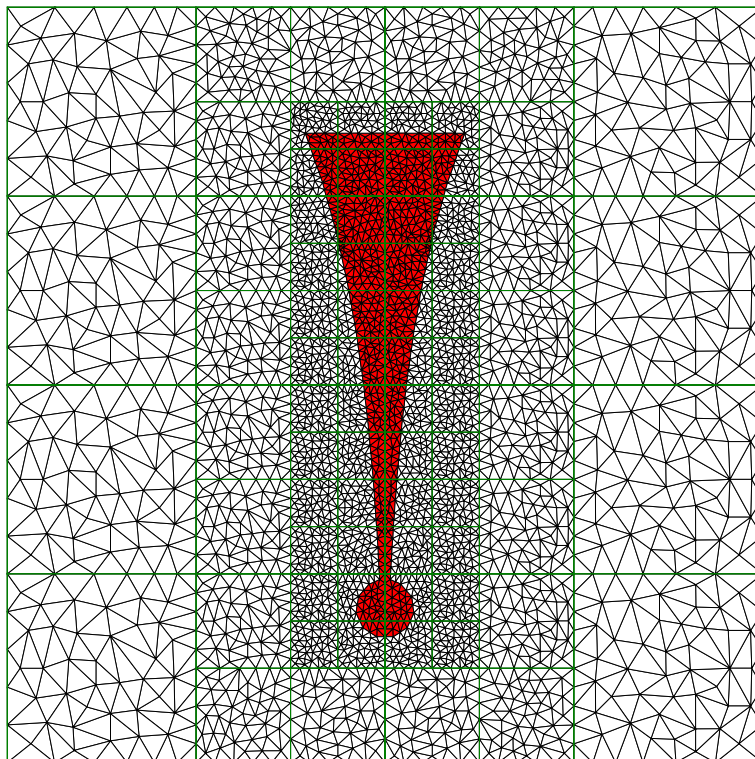


(a) Triangulation with  $fMCP=4$  and no merges. Generated 2162 triangles in 0.535 seconds.



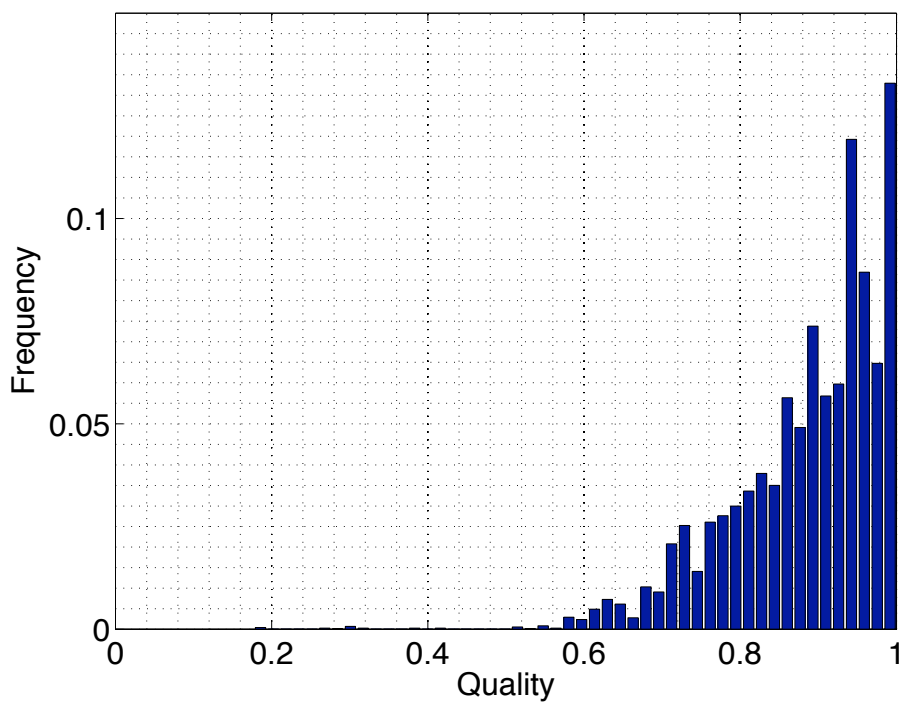
(b) Quality distribution.

Figure 3.24: Mesh of a Vivaldi flare using a Delaunay refinement algorithm as the successive triangulation procedure.



(a) Triangulation with  $fMCP=4$  and no merges. Generated 7168 triangles in 0.913 seconds.

### Vivaldi Flare, Relative, MCP 4



(b) Quality distribution.

Figure 3.25: Graded mesh of a Vivaldi flare using a Delaunay refinement algorithm as the successive triangulation procedure.

## CHAPTER 4

### CONCLUSION AND FUTURE WORK

#### 4.1 Conclusion

In this research, a combined quadtree and Delaunay mesh generation method for two dimensional non-manifold geometries arising from finite element method modeling was proposed. The method first subdivides the geometric problem using a grid until each grid cell meets the mesh control parameter of the topological elements it contains. This subdivision process with the grid is referred to as spatial partitioning in this thesis. The successive triangulation follows after the spatial partitioning process. The successive triangulation uses any "edge-preserving" triangulation algorithm to triangulate each cell. The "edge-preserving" triangulation algorithms that were considered for the successive triangulation method in this thesis are a basic conforming Delaunay triangulation algorithm and a generic constrained Delaunay refinement algorithm. Finally, the mesh quality improvement is the last of the three. The mesh quality improvement eliminate poor quality element by merging nearby vertices.

This approach is implemented as a part of this research, and the mesh generator has the following features:

- The mesh generator is a fully automated with minimal supervision from the user.
- Users can control the refinement level around each geometric topological entity or any point in space.
- Users have the option to refine further with a Delaunay refinement algorithm.

- The mesh generator improved mesh quality (without lower bounds guarantees) by eliminating skinny triangles through merging the geometry to quadtree vertices, at the price of poorer geometric approximation.
  - Under some conditions, the mesh generator has a quality guarantee,  $\frac{m_t(1-m_t)\sqrt{3}}{2m_t^2-m_t+1}$ .
- The mesh generator can be used with a CAD that generates parametric curves, non-manifold geometries, and poor tolerance geometries.

The results of the proposed mesh generator on various geometric models have lead to the following conclusions:

- Increasing the mesh control parameter will result in better approximation for smooth curves.
- Increasing the mesh control parameter will recover some of the geometric approximation that is lost during merging.
- Increasing the mesh control parameter will resolve topological inconsistencies from merging.
- Having more discretized edges in model can lead to poor quality elements.
- The time complexity of this approach is  $O(n \log n)$ .

## 4.2 Future Work

This mesh generator offers many features to FEM users, but more functionalities can be added to the mesh generator to meet the needs of FEM users. These functionalities are:

1. Implementation of a parallel successive triangulation – The successive triangulation process is the most time-consuming part of this method when there are



large number of elements. Therefore, running this process in parallel will save users a great amount of time.

2. Point merge operation includes merging vertices to any nearby vertices – Some poor quality elements still result because the nearby vertices did not merge. These nearby vertices did not merge because it does not fall into any of the merge cases that were considered. A consideration when dealing with merging any nearby vertices is chain merging. For example, vertex  $v_1$  merges to vertex  $v_2$ , vertex  $v_2$  merges to vertex  $v_3$ , and vertex  $v_3$  merges to vertex  $v_4$ . Vertex  $v_1$  is effectively merged to  $v_4$ , but  $v_1$  is not within the merging distance of  $v_4$ .
3. Subdivision of quadrants occurs automatically where there are topological inconsistencies – Topological inconsistencies in this approach are caused by merging nearby vertices. Subdivision of quadrants where topological inconsistencies occur will resolve topological inconsistencies by undoing the merge because the vertices are no longer within the merging distance of one another.
4. Extension of the implementation to a fully three-dimensional tetrahedron mesh generator – A three-dimensional tetrahedron mesh generator is useful to FEM users because they may want to solve some three-dimensional problems. One of the major challenges in implementing a three-dimensional mesh generator is the debugging process, especially when the algorithm is more sophisticated. Unlike two-dimensional meshes, the mesh needs to be inspected at different camera angles, different camera positions, and different assemblies of the mesh for discrepancies that can be spotted visually. Additionally, some operations should be avoided because they are computationally expensive, such as intersections of two solids or intersections of a face and a solid.

## BIBLIOGRAPHY

- [1] C. Johnson, *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, 1987. [Online]. Available: <http://books.google.com/books?id=Lq6RQgAACAAJ>
- [2] G. Strang and G. Fix, *An analysis of the finite element method*, ser. Prentice-Hall series in automatic computation. Prentice-Hall, 1973. [Online]. Available: <http://books.google.com/books?id=VZRRAAAAMAAJ>
- [3] R. Livesley, *Finite elements: an introduction for engineers*. Cambridge University Press, 1983. [Online]. Available: <http://books.google.com/books?id=0Hk6AAAIAAJ>
- [4] Ang, *A Beginner's Course in Boundary Element Methods*. Parkland: Universal Publishers, 2007.
- [5] A. Aliabadi, *The Boundary Element Method*. New York: John Wiley, 2002.
- [6] *The Boundary Element Methods in Engineering*. New York: McGraw-Hill, 1994.
- [7] G. Beer, *The Boundary Element Method with Programming*. Berlin: Springer, 2008.
- [8] J. R. Shewchuk, "What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures," In Proc. of the 11th International Meshing Roundtable, Tech. Rep., 2002.
- [9] U. Naumann and O. Schenk, *Combinatorial Scientific Computing*. CRC Press, December 2011.
- [10] C. L. Lawson, "Contributions to the theory of linear least maximum approximations," Ph.D. dissertation, University of California, Los Angeles, 1961.
- [11] A. Bowyer, "Computing dirichlet tessellations." *Comput. J.*, pp. 162–166, 1981.
- [12] D. F. Watson, "Computing the n-dimensional tessellation with application to voronoi polytopes," *Comput. J.*, pp. 167–172, 1981.
- [13] J. R. Shewchuk, "Delaunay Refinement Mesh Generation," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997, available as Technical Report CMU-CS-97-137.
- [14] "CGAL, Computational Geometry Algorithms Library," <http://www.cgal.org>.

- [15] J. Ruppert, “A delaunay refinement algorithm for quality 2-dimensional mesh generation,” 1994.
- [16] J. R. Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator,” in *Applied Computational Geometry: Towards Geometric Engineering*, ser. Lecture Notes in Computer Science, M. C. Lin and D. Manocha, Eds. Springer-Verlag, May 1996, vol. 1148, pp. 203–222, from the First ACM Workshop on Applied Computational Geometry.
- [17] L. P. Chew, “Guaranteed-quality mesh generation for curved surfaces,” in *Proceedings of the ninth annual symposium on Computational geometry*, ser. SCG '93. New York, NY, USA: ACM, 1993, pp. 274–280. [Online]. Available: <http://doi.acm.org/10.1145/160985.161150>
- [18] J.-F. Lee and R. Dyczij-Edlinger, “Automatic mesh generation using a modified delaunay tessellation,” *Antennas and Propagation Magazine, IEEE*, vol. 39, no. 1, pp. 34–45, feb 1997.
- [19] R. Löhner, “Generation of three-dimensional unstructured grids by the advancing front method,” in *Proceedings of the 26th AIAA Aerospace Sciences Meeting*, 1988.
- [20] D. J. Mavriplis, “An advancing front delaunay triangulation algorithm designed for robustness.”
- [21] J. Schberl, “Netgen an advancing front 2d/3d-mesh generator based on abstract rules,” *Computing and Visualization in Science*, vol. 1, no. 1, pp. 41–52, 1997.
- [22] R. Loehner and P. Parikh, “Generation of three-dimensional unstructured grids by the advancing-front method,” *International Journal for Numerical Methods in Fluids*, vol. 8, no. 10, pp. 1135–1149, 1988. [Online]. Available: <http://doi.wiley.com/10.1002/flid.1650081003>
- [23] M. Bern, D. Eppstein, and J. Gilbert, “Provably good mesh generation,” *J. Comput. Syst. Sci.*, vol. 48, pp. 231–241, 1990.
- [24] W. J. Schroeder, “Geometric triangulations,” Ph.D. dissertation, Rensselaer Polytechnic Institute, 1991.
- [25] S. A. Mitchell and S. A. Vavasis, “Quality mesh generation in three dimensions,” 1992, pp. 212–221.
- [26] M. S. Shephard and M. K. Georges, “Automatic three-dimensional mesh generation by the finite octree technique,” *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 709–749, 1991.
- [27] P. J. Frey and L. MARECHAL, “Fast adaptive quadtree mesh generation,” in *in: Proceedings of the Seventh International Meshing Roundtable*, 1998, pp. 211–224.

- [28] S. A. Mitchell and S. A. Vavasis, “Quality mesh generation in higher dimensions,” 1996.
- [29] M. Yerry and M. Shephard, “A modified quadtree approach to finite element mesh generation,” *IEEE Comput. Graph. Appl.*, vol. 3, no. 1, pp. 39–46, Jan. 1983. [Online]. Available: <http://dx.doi.org/10.1109/MCG.1983.262997>
- [30] M. S. Shephard, H. L. de Cougny, R. M. O’Bara, and M. W. Beall, “Automatic grid generation using spatially based trees,” in *Handbook of Grid Generation*, J. F. Thompson, B. K. Soni, and N. P. Weatherill, Eds. Boca Raton, FL: CRC Press, 1999, ch. 15.
- [31] K. foa Tchou, C. Hirsch, and R. Schneiders, “Octree-based hexahedral mesh generation for viscous flow simulations,” 1997.
- [32] “Opencascade,” <http://www.opencascade.org/>.
- [33] S. Holland and M. Vouvakis, “The planar ultrawideband modular antenna (puma) array,” *Antennas and Propagation, IEEE Transactions on*, vol. 60, no. 1, pp. 130–140, jan. 2012.
- [34] C. Brunetti and R. Curtis, *Printed circuit techniques*, ser. National Bureau of Standards circular. U. S. Govt. Print. Off., 1947. [Online]. Available: <http://books.google.com/books?id=JA7KjwEACAAJ>
- [35] S. Holland and M. Vouvakis, “The banyan tree antenna array,” *Antennas and Propagation, IEEE Transactions on*, vol. 59, no. 11, pp. 4060–4070, nov. 2011.
- [36] D. H. Schaubert, W. Elsallal, S. Kasturi, A. O. Boryssenko, M. N. Vouvakis, and G. Paraschos, “Wide bandwidth arrays of vivaldi antennas,” in *Wideband, Multi-band Antennas and Arrays for Defence or Civil Applications, 2008 Institution of Engineering and Technology Seminar on*. IET, Mar. 2008, pp. 1–20. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=4510671](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4510671)