PREMIUM QUALITY STEINER TRIANGULATIONS

By

HALE ERTEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

© 2009 Hale Erten

To my family

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Alper Üngör, for all of his guidance, support and patience throughout my studies and research. I am really grateful that he introduced me to Computational Geometry and inspired me with his immense knowledge and great enthusiasm. It was a great privilege to work under his supervision.

I would also like to thank my committee members, Dr. Jay Gopalakrishnan, Dr. Jeffrey Ho, Dr. Sanjay Ranka, and Dr. Meera Sitharam, for all of their help and valuable suggestions.

I am grateful to all my teachers in University of Florida, from whom I have learned various important subjects.

I shall acknowledge that most of the work presented in this thesis has been appeared in various publications. The materials in Chapter 3 have been published in the proceedings of the SGP07: ACM SIGGRAPH/Eurographics Symposium on Geometry Processing [51] and later published in SIAM Journal of Scientific Computing [53]. The materials in Chapter 4, have been published in the proceedings of the 19th Canadian Computational Geometry Conference [50] and the 6th International Symposium on Voronoi Diagrams [54] and submitted to the journal Applied Numerical Mathematics.

I would like to thank all my financial supporters throughout my entire study. University of Florida Computer and Information Science and Engineering department and National Science Foundation have been funding my research and education for four years. I also thank University of Florida Office of Audit and Compliance Review and Anesthesiology department for their partial supports.

I was delighted to work with Dr. Patrick M. Knupp at Sandia National Laboratories and Dr. Kirk Haller at Solidworks Corporation during my summer internships in their groups and thankful for leading me to work on diverse research projects and giving me financial support. I also thank Dr. Jörg Peters for providing me the latter opportunity.

Additionally, I am thankful to all my lab mates and friends. In particular, I am grateful to Seniha Esen Yüksel for her great friendship and help in the past four years. I also thank Chunchun Zhao, Amit Dhurandhar, Cem Boyacı, Özlem Subakan for the stimulating discussions and their support during my studies and Anubhav Singh for his valuable comments.

Last but not the least, I would like to thank my parents, Muhterem and Nail Erten and my sisters, Selda, Ferah, Hande and Müjde Erten for their never-ending love, support and understanding.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS 4

LIST OF TABLES 8

LIST OF FIGURES 9

LIST OF SYMBOLS 12

ABSTRACT 13

CHAPTER

1 INTRODUCTION 15

1.1 Motivation 16
1.2 Previous Work 18
1.3 Problem Statement 20
1.4 Contributions of Research 22

2 BACKGROUND 25

2.1 Triangulations 25
2.1.1 Delaunay Triangulations 26
2.1.1.1 Constrained Delaunay triangulations 28
2.1.1.2 Conforming Delaunay triangulations 29
2.1.2 Voronoi Diagrams 30
2.1.3 Quality Triangulations 31
2.1.4 Steiner Triangulations 32
2.2 Steiner Quality Triangulation Methods 33
2.2.1 Delaunay Refinement 33
2.2.2 Quadtree-based Methods 35
2.2.3 Advancing Front Methods 36
2.2.4 Circle-based Methods 38
2.2.5 Smoothing (Mesh Improvement) Methods 38
2.3 Steiner Quality Triangulations with Angle Constraints 41
2.3.1 No Small Angle Triangulations 43
2.3.1.1 Delaunay refinement 43
2.3.1.2 Other refinement algorithms 45
2.3.2 No Large Angle Triangulations 45
2.3.3 Smoothing 47

3 TRIANGULATIONS WITH LOCALLY OPTIMAL STEINER POINTS 50

3.1 Introduction 50
3.1.1 Our Idea and Contribution 52

3.1.2 Motivation . . . 52
3.1.2.1 Termination problem . . . 53
3.1.2.2 Mesh size . . . 56
3.2 Locally Optimal Steiner Points . . . 56
3.3 Analysis . . . 60
3.4 Relocation and Refinement . . . 63
3.5 Experiments . . . 64
3.5.1 Implementation . . . 65
3.5.2 Data Sets . . . 66
3.5.3 Performance Comparison . . . 66
3.5.4 Results on Uniform Triangulations . . . 76
3.5.5 Comparison: Smoothing Methods . . . 84

4 TRIANGULATIONS WITHOUT SMALL AND LARGE ANGLES . . . 87

4.1 Introduction . . . 87
4.1.1 Our Idea and Contribution . . . 87
4.1.2 Motivation . . . 89
4.2 α-petal, γ-petal, α-wedge, γ-slab, $[\alpha, \gamma]$-slice . . . 90
4.3 Intersection of Slices and Free Vertex Relocation . . . 93
4.4 Experiments . . . 96
4.4.1 Implementation . . . 96
4.4.2 Data Sets and Output Visualization . . . 96
4.4.3 Performance . . . 97
4.4.4 Comparison: Previous Delaunay Refinement Algorithms . . . 106
4.4.5 Comparison: Smoothing Methods . . . 109
4.4.6 Results on Uniform Triangulations . . . 112

5 DISCUSSIONS . . . 114

5.1 aCute Software . . . 114
5.1.1 Software Website . . . 114
5.1.2 Download and Usage . . . 114
5.1.3 Comparison: Differences and Similarities with Triangle Software . . . 115
5.2 Future Work . . . 116

REFERENCES . . . 118

BIOGRAPHICAL SKETCH . . . 128

LIST OF TABLES

Table page

3-1 Performance of Algorithm 3 with respect to others 67

3-2 Number of different types of Steiner points used by different algorithms 68

3-3 Percentage values of different types of Steiner points used by Algorithm 3 . . . 71

4-1 A qualitative comparison of our method against the previous methods 87

4-2 Performance of the algorithm for four different angle ranges 103

4-3 Comparison of our method with the previous Delaunay refinement methods . . 105

LIST OF FIGURES

Figure page

1-1 Delaunay triangulation and a Steiner quality triangulation of a complex domain. 21

2-1 Types of input domains with possible triangulations 26

2-2 Delaunay triangulation empty circle property 27

2-3 Maximizing the minimum angle and minimizing the maximum angle 28

2-4 Delaunay triangulation and three-dimensional convex hull relationship 28

2-5 Constrained and conforming Delaunay triangulations 29

2-6 Voronoi diagram and the dual Delaunay triangulation of a point set 30

2-7 Steiner triangulations of different input domains 32

2-8 Output triangulations with different qualities 34

2-9 Unbalanced and balanced quadtrees 35

2-10 Advancing front flow 36

2-11 Output of different refinement algorithms 37

2-12 Circle-based triangulation example 38

2-13 Relocating a point in Laplacian smoothing 39

2-14 Applying smoothing 41

2-15 Types of Steiner points 44

2-16 Output of different types of refinement algorithms 46

2-17 Output of different types of smoothing algorithms 49

3-1 The original Delaunay refinement algorithm vs. our idea 51

3-2 Circumcenter insertion 53

3-3 Termination problem 55

3-4 Termination problem on a simple domain 56

3-5 Classification of types of Steiner points 58

3-6 Locally optimal Steiner point 59

3-7 Type II (A) and (B) 61

3-8 Relocating a free vertex of a bad triangle . . . 63
3-9 Output of the Algorithm 3 for Boeing and octagon data sets . . . 66
3-10 Output of the Algorithm 3 for Italy and butterfly data sets. . . . 69
3-11 Output of the Algorithm 3 for cooking lady, seahorse and China data sets . . . 70
3-12 Output size comparison plots for Cyprus and crab data sets . . . 73
3-13 Output size comparison plots for hexagon and New York view data sets . . . 74
3-14 Output size comparison plots for Boeing and Florida data sets . . . 75
3-15 Output size comparison example for Boeing data set . . . 77
3-16 Output size comparison example for alligator data set . . . 77
3-17 Output size comparison example for Florida data set . . . 78
3-18 Plot of average output size ratio . . . 78
3-19 Plot of running times . . . 79
3-20 Output triangulations with max area constraint for various data sets . . . 80
3-21 Output triangulations with max area constraint for Texas data set . . . 81
3-22 Extensive plot of the output performance ratio based on max area for Puerto Rico data set . . . 81
3-23 Plot of the output performance ratio based on max area . . . 82
3-24 Output size comparison with max area constraint . . . 83
3-25 Comparing smoothing algorithms with our method . . . 86
4-1 Acute triangulation of the domain Acute . . . 88
4-2 Feasible region slice . . . 91
4-3 Additional types of Steiner points . . . 92
4-4 Relocating a free vertex . . . 93
4-5 Non-convex intersection regions . . . 94
4-6 Approximate convex superset of a slice region . . . 94
4-7 Coloring scales . . . 97
4-8 Plot of output size for varying α and γ . . . 98

4-9 Plot of output size and time performance of the algorithm based on a fixed α and a variable γ . . . 99

4-10 Plot of output size and time performance of the algorithm based on a fixed γ and a variable α . . . 100

4-11 Output triangulations of China data set for various minimum and maximum angle constraints . . . 101

4-12 Output triangulations of goldfish data set for various minimum and maximum angle constraints . . . 102

4-13 Output of the classical Delaunay refinement method . . . 107

4-14 Output of our method based on maximum and minimum angle constraints . . . 108

4-15 Size comparison of all data sets . . . 109

4-16 Comparing output sizes and angle histograms of previous classical Delaunay refinement algorithm and ours . . . 110

4-17 Output triangulations of various algorithms . . . 111

4-18 Premium quality output triangulations for Boeing and hexagon input domains with the additional maximum area constraint . . . 112

4-19 Premium quality output triangulations for Turkey data set with different maximum area constraints . . . 113

5-1 Algorithmic flow of Triangle and aCute for buffalo data set . . . 117

LIST OF SYMBOLS, NOMENCLATURE, OR ABBREVIATIONS

PSLG	Planar straight line graph
CDT	Constrained Delaunay triangulation
CVT	Centroidal Voronoi tessellation
ODT	Optimal Delaunay triangulation
max-min	Maximizing minimum
min-max	Minimizing maximum
α	Constraint minimum angle
β	Theoretical bound on minimum angle
ζ	Practical bound on minimum angle
γ	Constraint maximum angle
∂	Boundary

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PREMIUM QUALITY STEINER TRIANGULATIONS

By

Hale Erten

December 2009

Chair: Alper Üngör
Major: Computer Engineering

A complex geometric domain is required to be partitioned into a small set of simple and high-quality geometric elements for accurate numerical analysis. Given a two-dimensional input domain, we consider its quality triangulations as the discretization of the domain into non-overlapping triangles while each satisfies certain quality constraints. Related to this, many researchers studied two major problems. The first one is to generate triangulations with a lower bound on small angles, while the second is to generate triangulations with an upper bound on large angles. An upper bound on large angles does not imply a non-trivial lower bound on small angles. Here, we propose novel ideas to generate premium quality triangulations with significantly better simultaneous small and large angle bounds than the bounds provided by the existing methods in practice.

Delaunay refinement is a well-known triangulation technique, which generates size-optimal good-quality Delaunay triangulations. Delaunay refinement algorithms aim to compute triangulations that have all angles at least α. However, in practice, these algorithms usually work for $\alpha \leq 34^\circ$ and fail to terminate for larger constraint angles.

We describe two novel ideas to improve the performance of Delaunay refinement algorithms. The first idea is an effective use of the Voronoi diagram and unifies previously suggested point insertion schemes together with a new strategy. The second idea which is, integration of a local relocation strategy into the refinement process, leads

to a strategy that has significantly better performance than performing these two tasks independently. In addition to providing the same theoretical guarantees as the previous algorithms, our approach generates smaller triangulations and generally terminates for constraint angles up to 42°. We also extend our ideas to accommodate simultaneous small and large angle constraints. Experimental study shows that our method works for small angles as high as 41°, and large angles as low as 81°. Hence, we provide a substantial improvement over the existing results, especially for applications where acute or non-obtuse triangulations without small angles are desired. Employing these ideas provide the first software for computing premium quality triangulations for complex geometric domains.

CHAPTER 1
INTRODUCTION

Triangulation is an important algorithmic and geometric problem that has been studied over the years from different aspects. Most of the scientific applications use real world objects with complex geometric structures. In order to numerically analyze them, subdivision into simpler geometric components is a key step. If we consider two-dimensional space, triangles are the basic simplices for decomposition whereas tetrahedra are the same for three-dimensions. Hence, triangulating a given point set or a polygonal region is a crucial subproblem for wide-range of applications. Most classical examples can be given from finite element analysis and graphics. In addition, almost every geometry related field seeks algorithms for triangulations, such as networking, bioinformatics, cartography, architectural design, numerical analysis and many other scientific and engineering applications [11, 13, 16, 21].

Engineering applications create triangulations (generate meshes) as the first step of finite element method for numerical computations. Complex objects become combination of simple elements satisfying well-defined criteria. Discretization is required for solving continuous problems, so that target functions can be approximated with piecewise polynomials on elements [13]. Accuracy of solutions regarding differential and integral equations as well as speed depend on the quality of the triangulation [18, 111, 112]. Thus, underlying elements should satisfy some size and shape properties, usually preferred to be close to equilateral triangles [95].

Industrial implementations are always looking for more efficient algorithms, while they need triangulations which are suitable for their specific applications to avoid computational errors. Since different applications expect different quality metrics from triangulations, number of related problems increases rapidly. For instance, they could be looking for various constraints such as angle, weight, area, edge length, height and so on [5, 11, 95]. In this context, Babuška and Aziz [6] showed that large angles but

not necessarily small angles are bad for minimizing the gradient interpolation error. However, Shewchuk [105] pointed out that large angles and small angles are conflicting quality measures and showed that small angles are bad for minimizing the condition number of stiffness matrix, although large angles are acceptable. Since triangulations are necessary for solving wide-range of engineering and scientific problems, even a small progress on algorithms would be significant towards getting more accurate results on numerical calculations while saving enormous amount of time and money.

1.1 Motivation

While triangulations find use in many application domains, our primary source of motivation to study quality triangulations is engineering applications. In particular, we address the geometric modeling challenges raised by current and next-generation scientific computing algorithms. With the recent progress on numerical methods, geometric modeling remains to be a bottleneck in many numerical simulation applications. Here, we highlight the key challenges in this context.

Quality. Numerical methods that employ triangulations usually impose various quality constraints in the shape and size of the triangles. In general, the higher the quality of the triangulation is, the better the performance (accuracy, convergence speed, etc.) of the numerical method [6, 11, 105]. So, significant improvements over the existing triangulation algorithms and software addressing the traditional quality criteria, such as smallest angle, are always welcome. In some cases, the quality constraint becomes an absolute necessity as it translates to the theoretical guarantees on the numerical algorithm [105, 119]. As new numerical methods (are developed) addressing the needs of emerging application, there is greater need for triangulation algorithms and software providing various new quality criteria guarantee.

Triangulation size and speed. The size of a triangulation, that is the number of triangles, is a key factor in the running time of the application algorithm which uses the triangulation. Among two triangulations with the same quality bound, the one with the

fewer elements is preferred almost exclusively [68, 69]. One of the goals in this study is to reduce the output size of the triangulation algorithm. This also helps us improve the speed of the triangulation process. As a result, we have a double-gain in improving the numerical simulation time. Such an improvement in some applications may be the key performing real-time simulations.

Software. In addition to design of algorithms and heuristics, having their robust implementations in the form of popular software is essential for industrial and scientific practice. Among several different methods, a method can be distinguished by its strong contributions and its suitable software implementation. Consequently, engineering applications will consider using this method as a part of their numerical computations. However, robustness is very important [40]. Otherwise, even the significant benefits of the method could be ignored. Furthermore, improving an existing robust implementation of a popular algorithm will automatically have an effect on related applications.

Alternative quality criteria. Decomposing complex structures into triangles (triangulation) satisfying certain quality criteria is an integral part of engineering applications. As a result, they are always looking for better algorithms inducing more accuracy by improving quality criteria of triangulations. For majority of numerical applications, quality criterion is related to angles within a triangulation. Not only the smallest angle, but also the largest angle has significant importance for applications [36, 71, 96, 119]. Larger smallest angle and smaller largest angle triangulations are preferred to others. Moreover, in some cases acuteness of triangulation is necessary for efficiency of algorithms [119]. In addition to angle, total edge lengths of a triangulation or minimum triangle area in a triangulation also plays an important role. Thus, types of quality criteria is changing according to relevant engineering application.

Dynamic meshing. Numerical simulation of complex domains often require geometric modeling (triangulation) of an evolving geometry. The triangulation needs to adopt both the changing domain boundary, and the changing numerical conditions. In

general, this process involves an initial construction of geometric model of the domain followed by several iterations of numerical solutions. Throughout, the model can be updated in a straight-forward manner based on the numerical solution. After a while, however, a complete reconstruction of the triangulation becomes necessary as the geometric quality no longer assures an accurate simulation. Starting with a high quality triangulation, in this scenario, is particularly important as it can help to improve the efficiency of the application by allowing a faster, more accurate and longer simulation.

1.2 Previous Work

Triangulations have been studied since 1970s, and significant progress has been made. As the engineering applications need triangulations with certain quality measures, optimal triangulations became popular and different quality measures have emerged. Bern and Eppstein did an extensive survey about optimal triangulations for both two and three dimensional space [11]. In addition, other important surveys have also been completed [5, 16, 43, 44, 104, 116].

Decomposing a polygonal region into pieces (triangles) by itself is useful, without even considering any optimization/quality constraint. It has several practical applications for solving well-known real life problems [33]. However, majority of contemporary practical methods focus on quality triangulations, since approximation errors of an application depend on quality measurements of triangulations [6]. In particular, Delaunay triangulations attract many researchers due to their concurrent optimization of several quality measures and them being planar duals of Voronoi diagrams [26, 28, 29, 31, 68, 69, 84, 99, 103].

Quality triangulation problems usually require adding new points, called the Steiner points, to the input domains in order to improve quality and satisfy the given quality constraints. Throughout the years, Steiner quality triangulations have been generated by several different approaches, such as grid-based, circle-based, quadtree-based and other approaches [7, 13–15, 27, 48, 88]. In general, we call them refinement techniques,

such that an initial domain refined by additional points. Figure 1-1 gives the Delaunay triangulation and a Steiner quality triangulation of a complex geometric domain.

Among these approaches, in particular for generating Steiner quality triangulations with a minimum angle criterion, Delaunay-based and quadtree-based methods became widespread in use. Delaunay refinement methods insert Steiner points depending on the underlying initial Delaunay triangulation [28, 29, 45, 99, 118]. On the other hand, quadtree-based methods use well-known data structures, quadtrees, to find a location for a new Steiner point [13, 48, 88, 124]. (Refer Section 2.2.1 and Section 2.2.2 for more information.) These techniques have been foundations for essential software products. For example, Triangle [106] software implements several Delaunay refinement algorithms [101, 102], whereas another triangulation software QMG [89] implements quadtree refinement for mesh generation [88]. Quadtrees cause larger output triangulations and are limited for some quality measures in comparison to Delaunay-based methods. As a result, Delaunay refinement approaches are usually preferred over quadtree refinement approaches. Therefore, this affects the frequency amount in usage of Triangle software which is higher than QMG.

Many researchers have introduced ideas to improve Delaunay refinement algorithms from different aspects, like smallest angle bound, output size, time performance, and other important efficiency criteria [28, 45, 51, 68, 99, 118].

Largest angle criterion is known to be essential for many applications, as well as smallest angle. Having non-obtuse or acute triangulations could be mandatory for some algorithms or they might lead to more efficient versions of the algorithms, like surface interpolation, and mesh generation algorithms [11, 119]. Although there has been some work on non-obtuse triangulations [7, 12, 13, 15, 49] and acute triangulations [13, 22, 67, 70, 79, 120], a general algorithm for no small no large angle or acute triangulations has not been proposed. Moreover, the existing algorithms remain limited in their

scope and applicability. To our knowledge, there is no implementation or software that computes acute/non-obtuse and/or premium quality triangulations.

Different types of post-processing methods can also be used for mesh improvement; mainly edge flipping and smoothing operations [11]. After having an initial triangulation of a given input domain, it can be improved in terms of quality by flipping edges or adjusting mesh vertices which are allowed to move. In particular, smoothing could be done by simple computations, as in Laplacian smoothing, or it could require expensive optimization techniques applied on the triangulation globally or locally [3, 16, 46, 56, 125]. In addition, few studies combined refinement methods with local smoothing, however these are complex and inefficient combinations [32, 117].

Heuristic methods including polygonal decomposition [12], advancing front techniques [11, 17, 73, 77, 78, 81] and others possess various advantages depending on the problem to be solved [11, 75]. However, they remain less popular for quality triangulations compared to other refinement methods, unless integrated with other effective techniques [75, 118]. Further previous work details are given in Chapter 2.

1.3 Problem Statement

Quality triangulation concept contains a broad range of computationally challenging problems. Among these fundamentally hard problems, we studied two in detail. The traditional quality triangulation problem requires satisfying a lower bound on the smallest angle in a triangulation. The challenge is to provide termination and improve the best results which are relying on the Delaunay refinement process. Moreover, although smallest angle gives a trivial upper bound on largest angle, generating triangulations seeking non-trivial large angle bounds is also difficult and forms the second problem. These problems can be generalized as $[\alpha, \gamma]$-triangulation problem, where all the angles in a triangulation lie in the $[\alpha, \gamma]$ range. We refer them as premium quality Steiner triangulations having exceptional $[\alpha, \gamma]$ ranges. Eventually, this can provide high accuracy in engineering calculations.

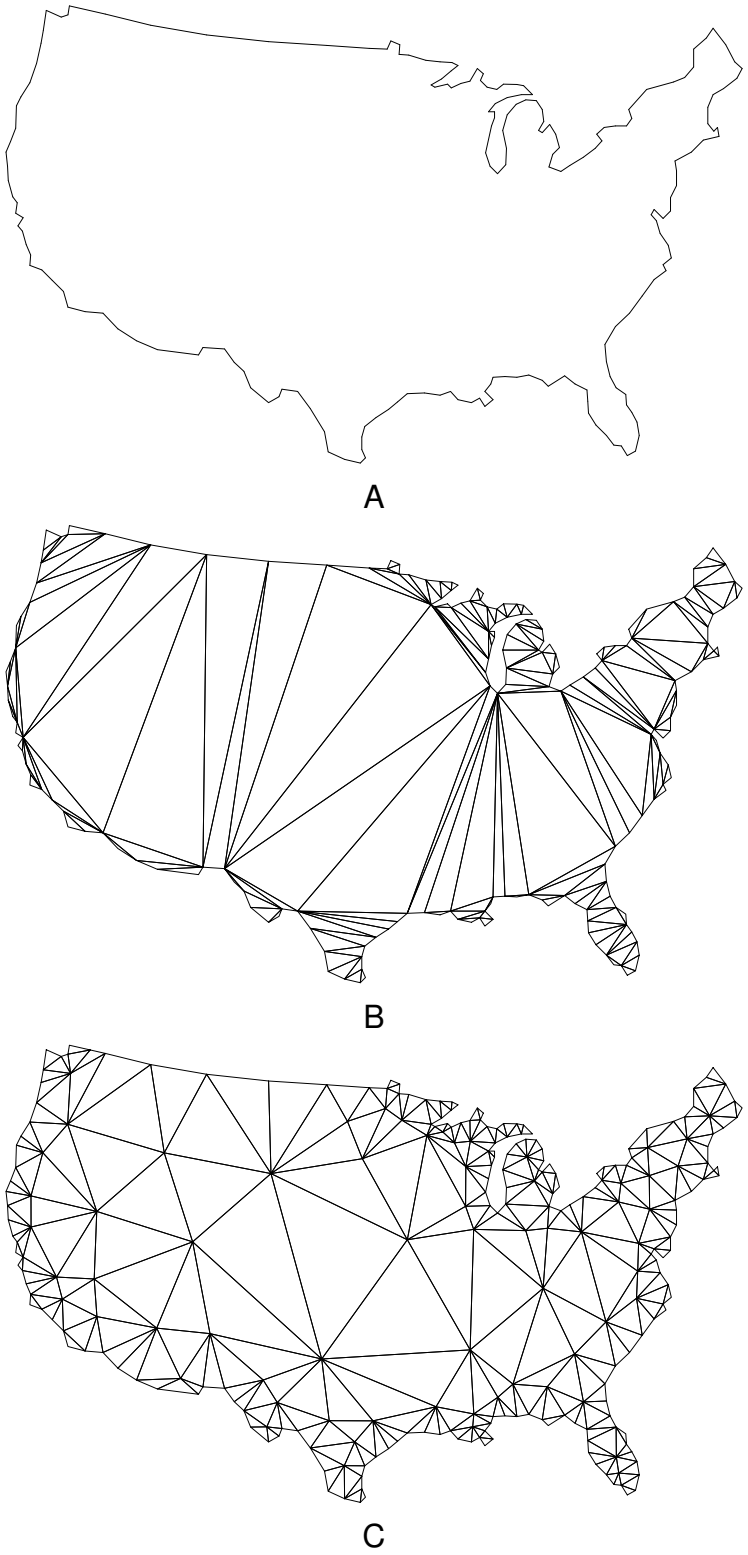


Figure 1-1. A) U.S.A. input domain. B) Delaunay triangulation of the domain. C) Steiner quality triangulation of the domain having all angles $> 20°$.

1.4 Contributions of Research

As a result of our motivation, we study quality triangulations. In particular, a well-known algorithm for generating quality triangulations, Delaunay refinement, is modified to contribute several important results for engineering applications. In this section, we review these contributions, while Chapter 3 and Chapter 4 present detailed explanations of the methods and achievements.

Unification of Steiner points. We give a simple definition for Steiner points which unifies the previous Steiner point insertion strategies [28, 45, 99, 104, 118]. This definition follows from a novel use of the Voronoi diagram, which is dual to Delaunay triangulation. The new rule automatically selects one of the four types of Steiner points, three of which (circumcenters, sinks, off-centers) are previously studied. We also extend this definition for generating $[\alpha, \gamma]$-triangulations. This new application of Voronoi structure could inspire solutions for other research problems.

Smoothing and refinement. Mesh refinement, i.e., carefully adding extra points [13, 28, 98, 99], and mesh smoothing, i.e., adjusting mesh vertices [3, 35, 60], are two important techniques for improving the quality of triangulations. These two methods are usually used one after the other (smoothing followed by refinement) [46] but rarely integrated [32, 76, 117]. We propose a simple framework where a simple local smoothing (point relocation) strategy is integrated into the refinement algorithm. It is certain that many other applications could benefit from this approach. Note that, our methods are more powerful than applying post-smoothing on a good-quality triangulation to provide premium quality triangulations.

Premium quality small angle bounds. With the new Steiner point definition and the integration of refinement and smoothing, we extend the practical angle bound of the Delaunay refinement algorithms. While the previous Delaunay refinement algorithms fail to compute meshes, for constraint angles as small as $33.9°$, the new algorithm computes good triangulations for constraint angles as high as $42°$. We extend the practical

constraint angle bound ζ by about 8°. This is a substantial improvement, especially considering the premium quality angle values that are provided. That is, for most of the applications having small angles close to 45° makes the important difference in accuracy and efficiency.

Premium quality non-trivial simultaneous large angle bounds. Our proposed method is the first that keeps all angles in an interval as narrow as $[40^\circ, 86^\circ]$, which is a substantial improvement over the existing results, especially for applications where acute or non-obtuse triangulations with no small angles are desired. The practical upper bound γ can be brought all the way down to 81° whenever 27° is an acceptable lower bound.

Sparse well-spaced point sets. Experiments show that our algorithm generating no small angle triangulations computes significantly smaller-sized triangulations than previous versions. The key idea in our Steiner point selection method is to stay away from the existing vertices. It turns out that traditional circumcenter insertion is not the best choice for this objective. Indeed, experiments give a surprisingly low percentage use for circumcenters among the four alternatives. We pick locations that result in longer edge lengths for the triangles of the mesh. In other words, the algorithm achieves creating sparse well-spaced point sets. In addition, despite the additional large angle constraint, output size of our algorithm generating triangulations without small and large angles tends to be similar to the output size of Delaunay refinement algorithms.

Software. Major performance contributions of modified versions of the Delaunay refinement algorithm inspired us to implement a new version of the popular triangulation software, Triangle. Thus, we are helping to improve applications relying on this software program. Furthermore, our heuristic method on no small no large angle triangulation problem leads to a completely new robust software generating premium quality triangulations and satisfying the need for this type of triangulators. Also, the proposed methods are mesh refinement methods relying on Delaunay triangulations and

they run faster in practice than the previous Delaunay refinement algorithms and much faster than the iterative mesh smoothing methods. This is mainly because the proposed algorithms introduce fewer Steiner points than others. It is important to note that our algorithms have more components than the previous ones; the local smoothing routine and the more complex point insertion routine. However, the simplicity of the proposed ideas enables a faster implementation overall.

Handling other quality criteria. Our methods also allow other constraints such as the largest triangle area constraint in conjunction with smallest (and/or largest) angle constraint simultaneously. Hence, they can be used for computing graded or uniform triangulations for generating premium quality triangulations.

Handling complex geometry. Our methods still work for complex geometric input domains; a property many acute and non-obtuse triangulation algorithms including some smoothing methods lack in particular.

CHAPTER 2
BACKGROUND

As a preliminary to the rest of the thesis, we give a formal background information on quality triangulations in the plane, and in particular Steiner triangulations with angle constraints. The basic input domain types for a triangulation problem are given [11]:

- **Point set:** The input domain is a set of points, so that the resulting triangulation always contains the convex hull of the set as a boundary.
- **Simple polygon:** The input domain is a simple polygonal region bounded by a closed curve.
- **Polygon with holes:** The input domain is a simple polygonal region bounded by several disjoint closed curves.
- **Planar straight line graph (PSLG):** The input domain is a set of points V and a set of non-crossing edges E. The triangulation of this type of input must contain the edges inside E as part of the triangulation.

Note that, PSLG is the most general definition, where the other mentioned types of inputs can be represented in terms of PSLGs. Also, triangulations can be seen as PSLGs in a special way (see Figure 1-1).

2.1 Triangulations

Definition 1. *Given a set of points S, a triangulation is a planar geometric graph $T = (S, E)$, where the set of edges E is maximal. Given a simple polygon or a planar straight line graph $G = (S, E)$, a triangulation is a planar geometric graph $T = (S, E \cup E')$ where G is augmented by an edge set E' until it is a triangulation.*

In order to generate triangulations, the most straight-forward algorithm is to pick non-crossing diagonals, where a diagonal is a new edge connecting two vertices of the polygon, which is entirely inside the polygon. This can be done in $O(n^2)$-time, however, this time-bound is later defeated by $O(nlogn)$ [62], $O(nloglogn)$ [115] and $O(n)$ [23] time algorithms regarding simple polygonal domains. For slightly modified versions of the same problem, different results have been obtained. For instance, some of the variations are proven to be NP-complete [11].

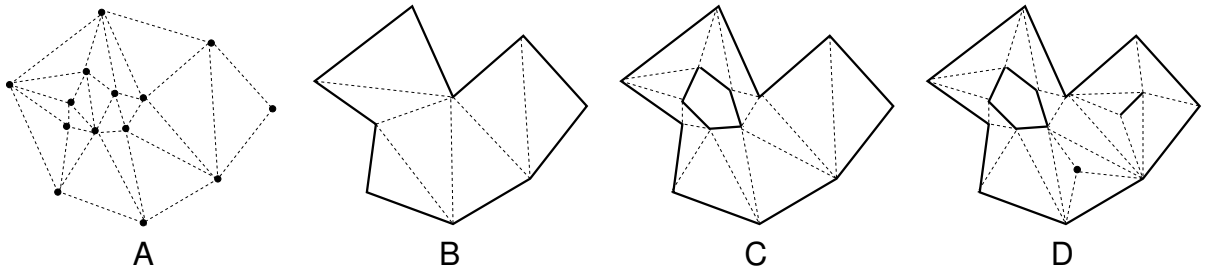


Figure 2-1. Types of input domains with possible triangulations in dashed lines. A) Point set. B) Simple polygon. C) Polygon with a hole. D) PSLG.

2.1.1 Delaunay Triangulations

Definition 2. *Given a set of points P in the plane, Delaunay triangulation of P, denoted by $DT(P)$, is a triangulation of the given domain P, such that no point in P is inside the circumcircle of any triangle in $DT(P)$.*

Definition 3. *Given three non-collinear points p_1, p_2, p_3 in the plane, the circumcircle of the triangle on these points is the unique circle passing through all three. The circumcenter is the center of the circumcircle.*

Delaunay triangulations are distinguished by their duality relation with Voronoi diagrams and being optimal triangulations for many quality measures, e.g., max-min angle triangulations. In addition, since there exist numerous efficient algorithms for generating Delaunay triangulations, this special type of triangulation has become widely popular, especially in engineering applications.

The empty circle property (see Figure 2-2) makes Delaunay triangulations the dual of Voronoi Diagrams. More importantly, the minimum angle of the triangulation is maximized among all other triangulations of a given two-dimensional point set [108]. See Figure 2-3.

Many $O(nlogn)$-time algorithms have been introduced for finding Delaunay triangulations in different ways. The list contains convex hull based algorithms [97], divide and conquer algorithms [41, 65, 74], plane-sweep algorithms (which generate Voronoi diagrams) [57, 58] and randomized incremental algorithms [64].

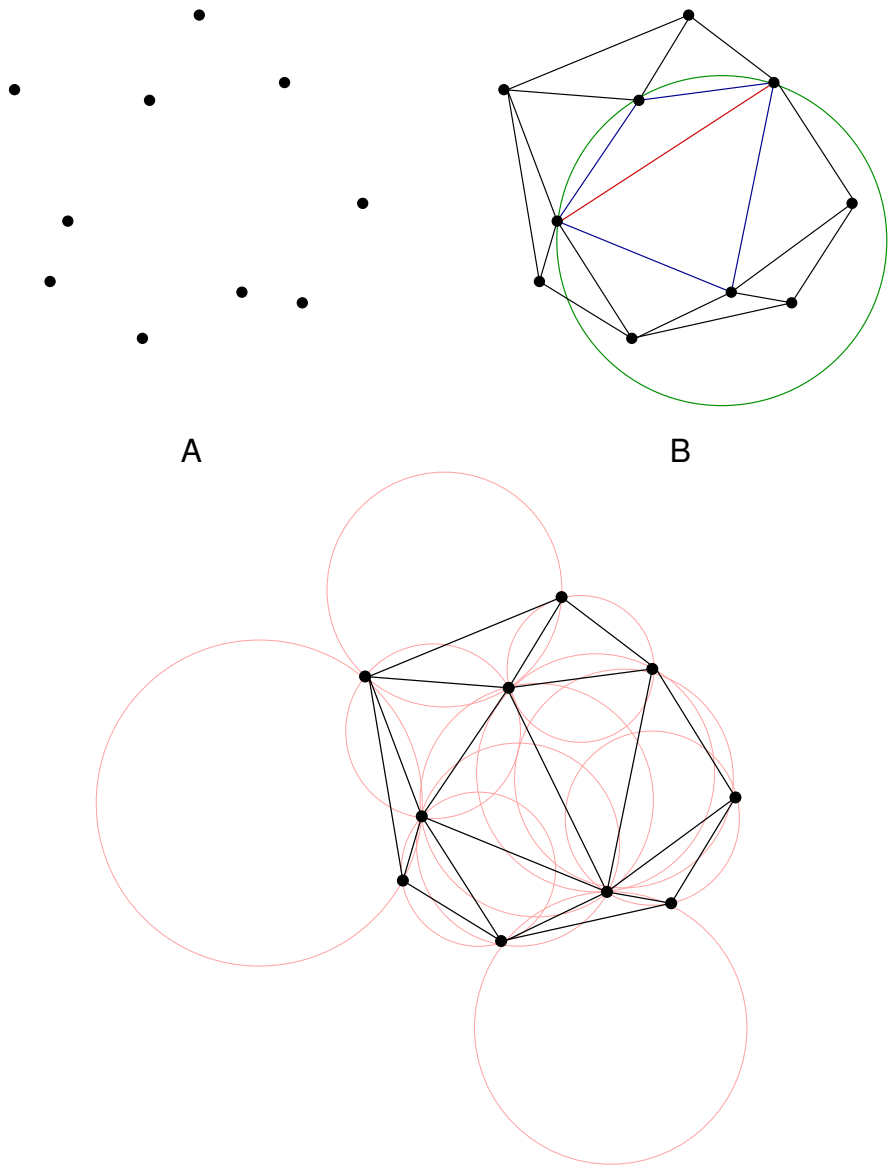


Figure 2-2. Delaunay triangulation empty circle property: The circumcircle of any triangle in the Delaunay triangulation is empty (contains no vertices of P). A) Input point set P. B) Not a Delaunay triangulation (empty circle property is violated). C) Delaunay triangulation.

In practice, these methods have been improved by several studies. Moreover, parallel versions have been proposed to provide more efficient implementations [30, 110]. A detailed survey and a comparative study on Delaunay triangulations can be found in [58, 113].

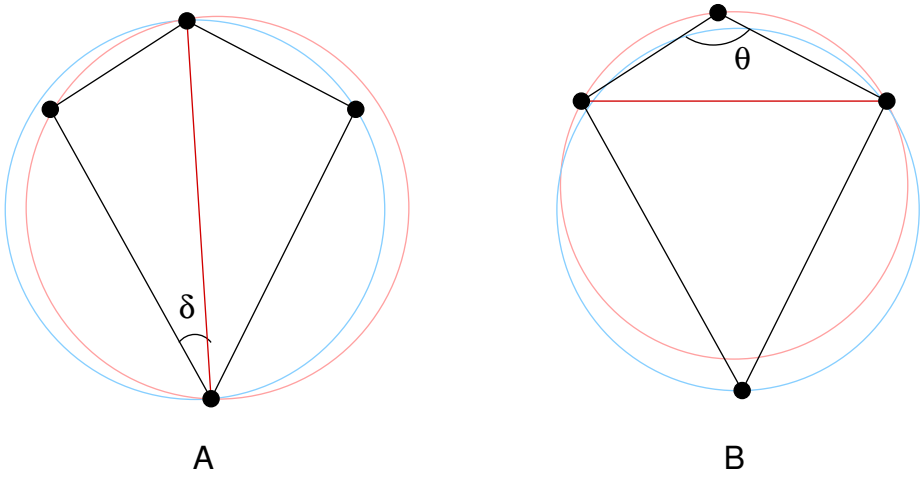


Figure 2-3. Maximizing the minimum angle and minimizing the maximum angle. θ denotes the maximum angle, δ denotes the minimum angle in the triangulation. A) Not a locally optimal edge (not a Delaunay triangulation, but the maximum angle is minimized). B) A locally optimal edge (Delaunay triangulation, the minimum angle is maximized, but not vice versa.).

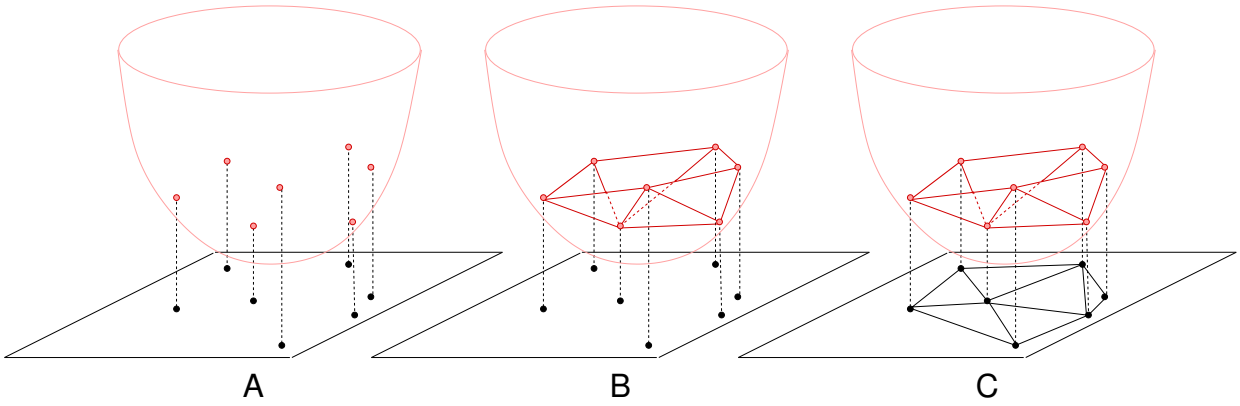


Figure 2-4. In convex hull based algorithms, a mapping between the input vertices and three-dimensional space is established as a paraboloid. Then, the lower three-dimensional convex hull of the paraboloid is projected onto the plane to obtain the Delaunay triangulation of the point set.

2.1.1.1 Constrained Delaunay triangulations

Definition 4. *A constrained Delaunay triangulation (CDT) contains an edge* ab *between two input vertices, if and only if* a *is visible to* b*, and some circle through* a *and* b *contains no input point* c *visible to segment* ab*.*

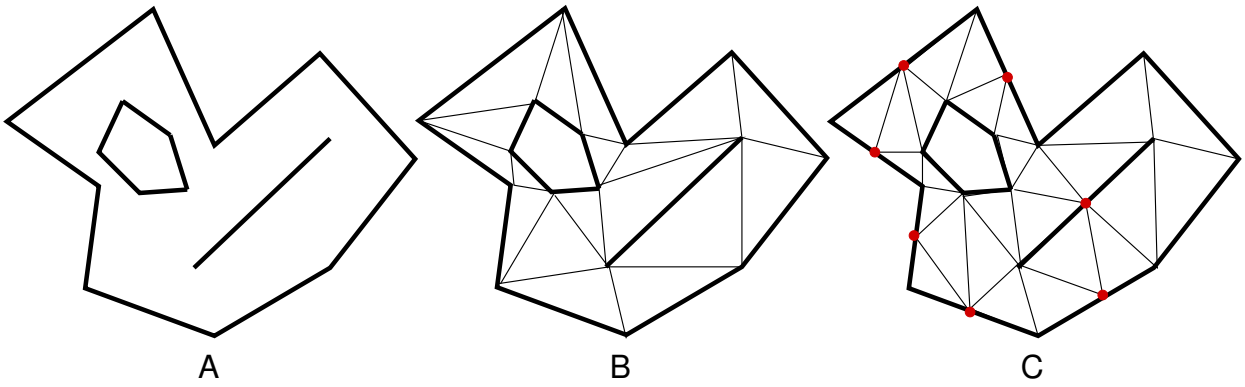


Figure 2-5. Constrained and conforming Delaunay triangulations of the given PSLG. A) Input PSLG. B) Constrained Delaunay triangulation. C) Conforming Delaunay triangulation.

Definition 5. *Point a is visible to point b if line segment ab does not cross any edge of the graph (ab may intersect an edge without crossing it). Point a is visible to an edge segment bc if it is visible to some point on bc* [11].

For PSLG type of input domains, Delaunay triangulations are defined as constrained, since the given edges are required to be preserved as a single edge in the triangulation. Note that, a constrained Delaunay triangulation might contain triangles that are not Delaunay. However, they satisfy the constrained Delaunay condition (see Figure 2-5 (B)). After a simple $O(n^2)$-time edge flipping algorithm, Chew proposed an $O(nlogn)$-time divide and conquer algorithm for finding CDTs in general [31]. For convex polygons, Aggarwal et al. gave a linear-time algorithm [1].

2.1.1.2 Conforming Delaunay triangulations

Definition 6. *A conforming Delaunay triangulation allows each PSLG input segment to be subdivided into several edges by inserting additional points on the segment, such that the Delaunay property is satisfied.*

Conforming Delaunay triangulations are Delaunay triangulations with Steiner points (see Figure 2-5 (C)). Note that, Steiner points are generally used for improving the mesh quality. Hence, algorithms employing Delaunay triangulation can be seen as

generating good quality conforming triangulations, such as Chew's [28] and Ruppert's [99] Delaunay refinement algorithms.

2.1.2 Voronoi Diagrams

Definition 7. *Let S be a set of n sites in Euclidean space of dimension d. For each site p of S, the Voronoi cell $Vor(p)$ of p is the set of points that are closer to p than to the other sites of S. The Voronoi diagram $Vor(S)$ subdivides the space into partitions induced by Voronoi cells.*

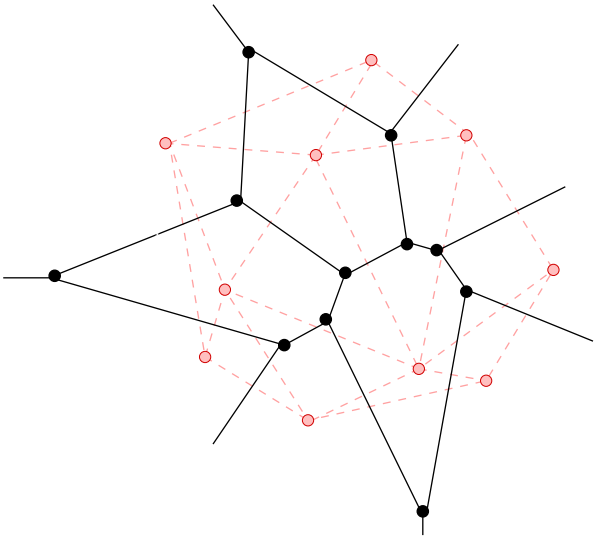


Figure 2-6. Voronoi diagram (solid lines) and the dual Delaunay triangulation (dashed lines) of point set given in Figure 2-2.

Voronoi diagrams are used for solving variety of problems. In particular, nearest neighbor queries have numerous applications, such as location-database applications, data mining problems as well as polymer physics topics [4, 11, 69]. Convex hull problem and Delaunay triangulation problems can be reduced to Voronoi diagrams. Thus, algorithms generating Voronoi diagrams are also applicable for Delaunay triangulations and vice versa. Simply, Voronoi diagrams can be generated in $O(n^2 logn)$-time by using half plane intersection algorithm. However, $O(nlogn)$-time algorithms are known for finding the Delaunay triangulation of a point set. Hence, this must be true for Voronoi diagrams as well. Indeed, Fortune's $O(nlogn)$-time sweepline algorithm turns out to be

the optimum algorithm for Voronoi diagrams [57, 58]. An extensive survey on Voronoi diagrams can be found in [4].

2.1.3 Quality Triangulations

Definition 8. *Given a point set or a PSLG, a quality triangulation constructs a triangulation T, such that for each triangle t of T, $quality(t)$ satisfies condition C.*

A quality condition can be defined as a threshold value τ based on a triangle property, such as angle, area, edge length and others. So that, this threshold value needs to be satisfied according to the condition, like smaller than τ, larger than τ, and similar. For example, having a quality triangulation with the smallest angle larger than an angle threshold value, such as $\tau = 30°$, is a common problem.

Optimal and quality triangulations are strongly related problems with a small distinction. The former optimizes a quality measure by obeying quality conditions implicitly. For instance, maximizing the smallest angle of the triangulation problem is same as having a quality triangulation satisfying the condition of having the smallest angle at least as much as the optimal maximum angle value. However, a quality triangulation is not necessarily an optimal triangulation and both have significant importance.

Algorithms for optimal triangulations contain divide and conquer [100], convex hull [97], sweepline [57, 58], and randomized incremental [64] approaches. In addition, edge flipping, edge insertion algorithms, greedy and dynamic programming methods are also used [11]. Certain variations remain still as open problems having only approximate or heuristic solutions, such as minimum weight triangulation. A detailed study on optimal triangulation algorithms can be found in [121].

Among many optimal triangulation problem, two of them are more relevant to our study than others. These are referred as max-min and min-max angle triangulations. As noted, Delaunay triangulations maximize the minimum angle in a triangulation and provide a direct solution to max-min angle triangulation problem. Although avoiding

small angles help to avoid large angles, Delaunay triangulations do not give min-max angle triangulations. (See Figure 2-3.) For finding such triangulations, edge flipping method has been shown to provide good results in practice [47, 63]. However, this technique might result in a local optimum rather than a global one. Hence, the computationally expensive edge insertion algorithm is introduced [10] and has been shown to exactly solve the min-max angle triangulation problem [47].

2.1.4 Steiner Triangulations

Definition 9. *A Steiner triangulation is any triangulation that adds additional sites to an existing triangulation to improve its measure of quality.*

In some cases, for a given input domain, its triangulation cannot reach the required level of quality. Therefore, a variant of quality triangulations is introduced allowing additional points in order to optimize or just satisfy the given quality measures/conditions [11] (see Figure 2-7). These additional points, as mentioned before, are called Steiner points. Although Steiner points might improve the quality, too many points can cause problems or inefficiency for several applications due to size. However, this new type of quality triangulation is suitable for many engineering problems allowing extra vertices, such as mesh generation of a structural object.

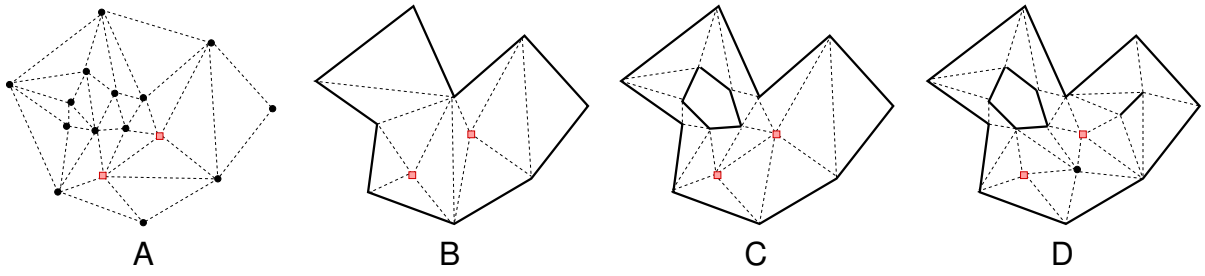


Figure 2-7. Types of input domains with possible Steiner triangulations in dashed lines. A) Point set. B) Simple polygon. C) Polygon with a hole. D) PSLG.

Note that, Steiner versions of quality triangulation problems need an additional goal as size optimality to avoid oversized triangulations, that is, there is a trade-off between quality and number of Steiner points. For example, the Delaunay triangulation

is known to be the optimal triangulation for maximizing the minimum angle condition [33, 63]. When Steiner points are allowed, the problem becomes a Steiner quality triangulation problem to find the smallest-sized triangulation with small angle less than a threshold value. However, this type of optimality makes the problems complex. Therefore, approximation algorithms providing quality triangulations using moderate number of Steiner points are introduced.

Most crucial part of the Steiner triangulation algorithms is to have an effective point insertion strategy. There are numerous strategies for locating Steiner points, including Delaunay-based, grid-based, circle-based and quadtree-based methods [7, 13–15, 27, 48, 88]. Section 2.2 describes these methods in detail.

2.2 Steiner Quality Triangulation Methods

In practice, three main methods are preferred to construct Steiner quality triangulations; Delaunay, quadtree and advancing front based techniques. These methods differ in terms of their strategy to find appropriate locations for Steiner points. Here, we also describe circle-based methods and smoothing techniques.

2.2.1 Delaunay Refinement

Employing Delaunay triangulations for generating Steiner quality triangulations was not discovered immediately. However, its special properties inspired researchers. In particular, since Delaunay triangulations maximize the minimum angle among other triangulations, this is considered as a huge advantage. Therefore, after pre-calculation of Steiner points by grid-based methods, Steiner quality triangulation of the pre-processed input domain is generated in Delaunay form [11].

Later, Frey introduced the idea of using Delaunay triangulation of the input and the new boundary vertices, and insert Steiner points inside the domain to eliminate bad triangles, i.e., non-quality triangles [61]. The basic strategy is to keep the Delaunay property by adding circumcenters of the bad triangles. Since this is an inexpensive local operation, Delaunay refinement algorithms became popular. Note that, the relationship

between the number of Steiner points and the quality of triangulation remains as a trade-off (see Figure 2-8).

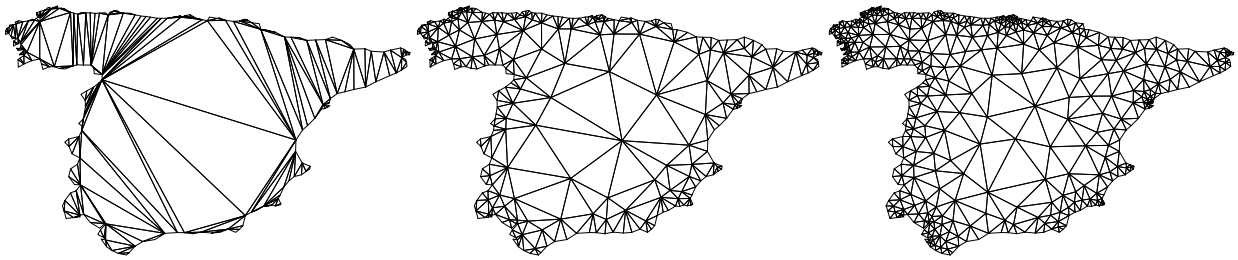


Figure 2-8. Output triangulations with different qualities (α: minimum angle). A) Delaunay triangulation: $\alpha = 1.35°$, 0 Steiner points. B) Steiner triangulation: $\alpha = 20°$, 106 Steiner points. C) Steiner triangulation: $\alpha = 30°$, 299 Steiner points.

Chew proposed a Delaunay refinement algorithm to obtain uniform density triangulations [28]. He simply proposed to insert nearby points in order to eliminate poor quality triangles where circumcenters of these triangles are used as the location for the new point [66]. Then, Ruppert introduced maybe the first Delaunay refinement algorithm which gives both size and shape guarantees [99] by following a similar idea. He proved that, the algorithm successfully generates size-optimal meshes with no angles smaller than around $20.7°$. Chew published another algorithm finding triangulations with size-optimality and no angles smaller than around $26.5°$ [29]. Without size-optimality guarantee, the angle boundary is advanced to $30°$ by a sphere packing argument. The classical Delaunay refinement algorithms can be generalized as in Algorithm 1.

However, the Delaunay refinement method still suffers from the so-called termination problem [53], such that the algorithm does not terminate for large angles. Pav also studied Delaunay Refinement algorithms in his thesis and pointed out some cases where the Delaunay refinement algorithm fails [93]. In practice, this method generates Steiner quality triangulations for minimum angle bound upto $34°$. As an alternative to circumcenters, mainly sinks [45] and off-centers [118] were introduced. Additional

information regarding quality Steiner triangulations with angle constraints is given in Section 2.3.

Algorithm 1 Delaunay Refinement Algorithm

Compute the Delaunay triangulation of the input
while $\exists$ a bad triangle pqr with shortest edge pq
  Insert a point x which is the circumcenter

2.2.2 Quadtree-based Methods

Definition 10. *A quadtree is a recursive partitioning of a region of the plane into axis-aligned squares. It is a tree whose nodes either are leaves (no child) or have four children. The root covers the entire region. A square can be divided into four child squares, splitted by horizontal and vertical line segments through its center* [11].

Grid-based methods for Steiner quality triangulations has been used to guarantee some quality measures [11]. For instance, in [7], Baker et al. used uniform grids to give bounds for the angles in the triangulation. Quadtrees can be seen as grids with levels (see Figure 2-9 (B)), which has been used to solve quality triangulation problems.

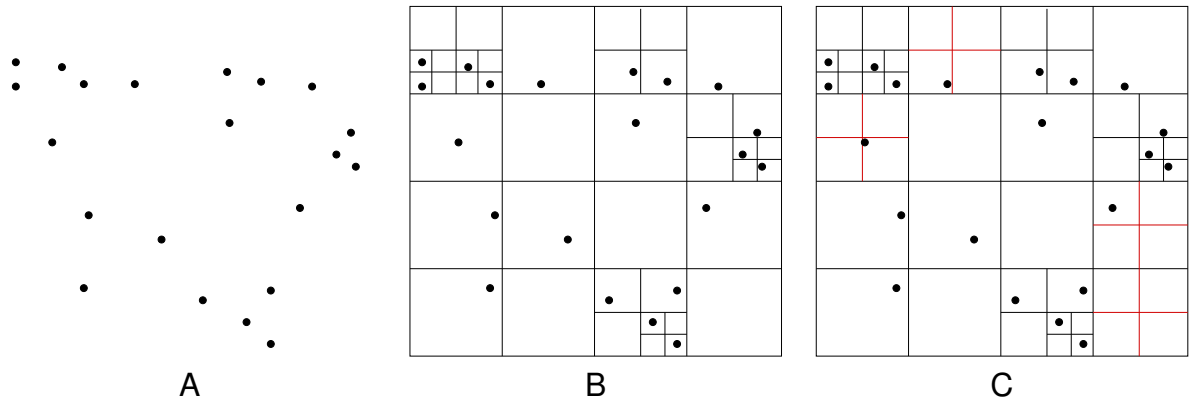


Figure 2-9. A) Input point set. B) Unbalanced quadtree. C) Balanced quadtree.

For Steiner quality triangulations, quadtree-based refinement algorithms insert Steiner points based on the quadtree formed from the input domain. They give triangulations within a constant factor of the optimal and satisfy various quality measures [13, 27, 48, 88, 124]. In addition, most of the algorithms use balanced

quadtree structure, which helps to satisfy the quality measures by the help of the proper grading information (see Figure 2-9 (C)).

Definition 11. *In a balanced quadtree, side lengths of any two horizontally or vertically adjacent leaf squares differ by at most a factor of* 2.

Although quadtree methods provide some guarantees for several triangulation problems, they also tend to generate large-sized triangulations by adding too many Steiner points. Moreover, usually they cannot be extended for PSLGs [102]. Also, they generate axis-aligned triangulations and have limitations on angles and on computing boundary elements [11].

2.2.3 Advancing Front Methods

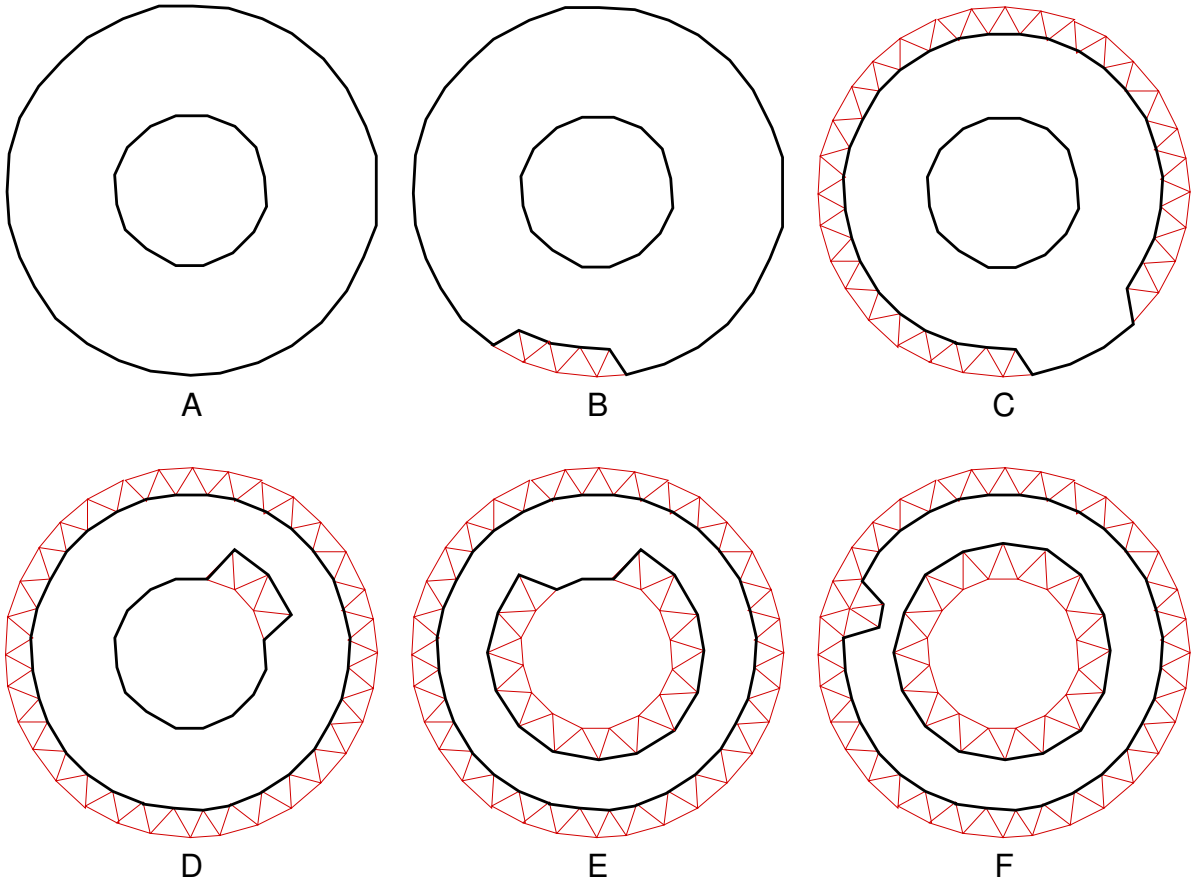


Figure 2-10. Steps of advancing front method are depicted.

Advancing front methods [11, 17, 73, 77, 78, 81] are suitable for most of the engineering applications, since the idea is relatively efficient and has an easy implementation compared to Delaunay and quadtree based methods. In addition, input boundary can be well-represented and the output meshes usually contain triangles with orientations which are appropriate for particular engineering problems, such as problems related to fluid dynamics [11]. However, in general advancing front methods do not guarantee neither size nor quality bounds. As an exception to this, the Delaunay refinement algorithm given in [118] can be seen as the first advancing front type of technique that provides theoretical guarantees.

Basic idea is to incrementally create new elements beginning from the input boundary and adjusting the elements along the way. In each step, an appropriate place is found for a Steiner point while keeping and updating the front. However, determining the point locations is a difficult task. Researchers tried different strategies, such as putting points on the pre-defined contour lines or using well-known curves for the point locations [11]. An illustration of advancing front technique is shown in Figure 2-10. In addition, Figure 3-9 gives a comparison between aforementioned triangulation methods. Note that, quadtree-based method inserts significantly larger number of points and generate axis-oriented triangulations.

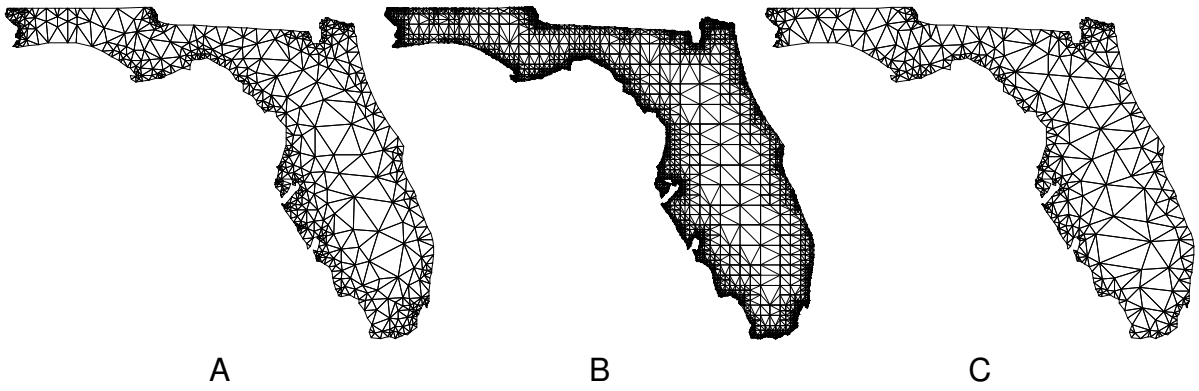


Figure 2-11. Output of different refinement algorithms. A) Delaunay Refinement. B) Quadtree Refinement. C) Advancing Front.

2.2.4 Circle-based Methods

In circle-based algorithms, the centers of the disks covering the input polygonal region is used to determine locations for Steiner points [15, 49, 87, 107]. Several quality constraints aimed to be satisfied using circle packings. For example, Bern et al. [15] and Eppstein [49] gave algorithms generating non-obtuse triangulations, where all the angles are no larger than $\pi/2$. Moreover, Shimada and Gossard [107] introduced bubble meshing, which uses the relationship between Voronoi diagram and tightly packed circles to generate quality triangulations. Miller et al. [87] also employed circle packings together with quadtrees and Delaunay triangulations to provide quality triangulations for numerical methods. Figure 2-12 gives an example circle packing and its triangulation.

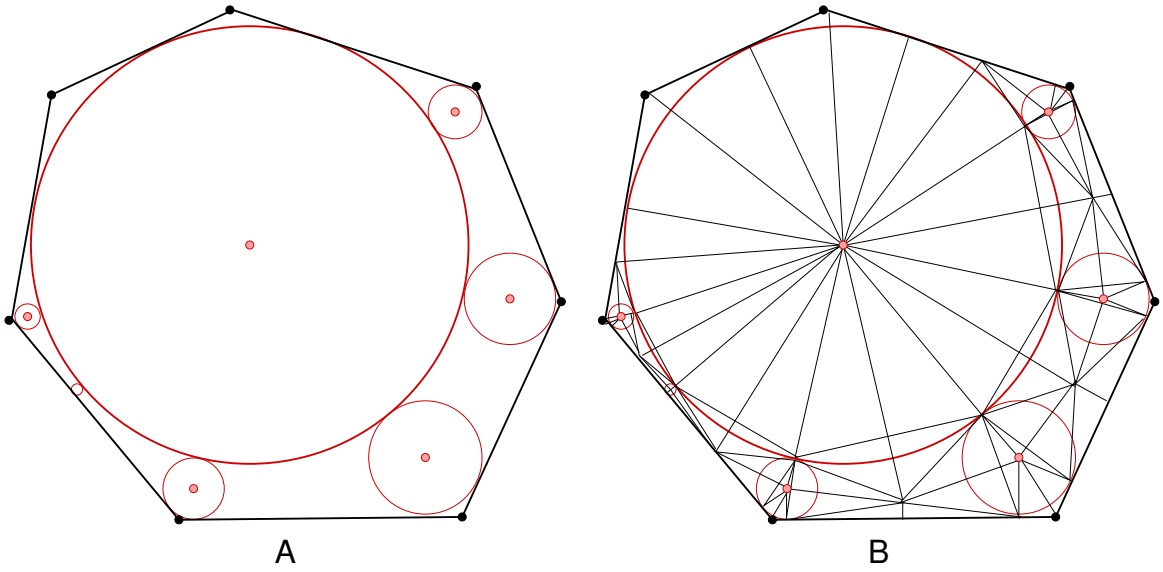


Figure 2-12. Circle-based triangulation example. A) Circle packing. B) Output triangulation.

2.2.5 Smoothing (Mesh Improvement) Methods

Although triangulation algorithms provide good quality, output triangulations usually can be further improved. In particular, Steiner quality triangulations mostly formed by the additional points, which are not part of the input domain and usually allowed to be moved. Relocating these free vertices is another way to improve the quality. This post-processing method is referred as smoothing or mesh optimization/improvement,

which is an important part of mesh generation to improve quality by adjusting (usually non-boundary) free vertices [11]. This can be also incorporated with potential edge flipping operations to obtain higher quality triangulations.

The idea is to perform local relocations and improve the overall shape or size quality of a triangulation. The relocation strategy can be based on an energy function or even a heuristic. There are many smoothing methods which are relying on different approaches, such as geometry, statistics and optimization [2, 3, 8, 16, 19, 20, 24, 39, 46, 56, 59, 91, 92, 120, 122, 123, 125].

Laplacian smoothing. A free vertex is simply relocated to the centroid of the vertices connected to that vertex [56] (see Figure 2-13). This technique is widely used due to its simplicity and effectiveness. However, it does not guarantee quality improvement where inverted elements might be generated. Therefore, many studies introduced different versions of Laplacian smoothing or combined with other methods. An overview of those can be found in [20].

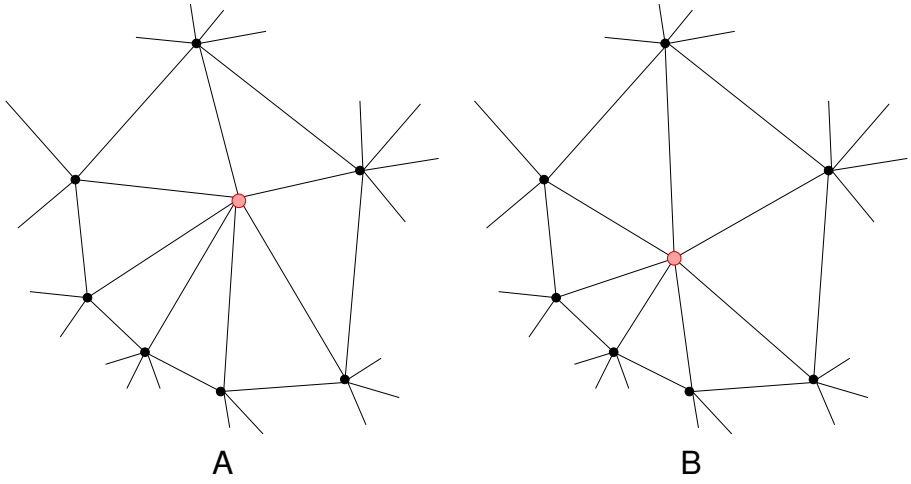


Figure 2-13. Relocating a point to the centroid of its neighbors. A) Before Laplacian smoothing. B) After applying Laplacian smoothing.

One of the variations of the standard Laplacian smoothing algorithm is the smart Laplacian smoothing [20, 59] which is also called constrained Laplacian smoothing. The main difference between the original and this variant is relocating points only if there

is an improvement in the neighborhood. Briefly, a vertex is relocated when the new candidate location improves the quality of the elements in the star. Otherwise, the node remains in its current location.

Centroidal Voronoi tessellation (CVT)-based smoothing. A Voronoi tessellation is called a centroidal Voronoi tessellation (CVT) when its generating points are the centroids of the corresponding Voronoi regions [37]. This special structure has been shown to have applications in many fields, such as image processing, clustering, cell division, and others. Du et al. discussed many properties and applications of CVT in [37–39, 122]. An immediate algorithm for generating such tessellations is to iteratively compute the Voronoi regions for each point and updating their locations to the centroid of each region until convergence. A discussion on other deterministic and probabilistic methods for CVT generation can be found in [37]. The iterative method is naturally suitable for mesh smoothing. Thus, researchers proposed different smoothing techniques based on centroidal Voronoi tessellation concept [2, 24, 39, 122].

Chen and Xu [25] studied Delaunay triangulations in terms of linear interpolation error for a given function and introduced optimal Delaunay triangulation (ODT), which is strongly related to CVTs. This helped Chen to design a new mesh smoothing strategy [24], ODT-based smoothing, which aims to equally distribute the edge lengths based on the function to be approximated.

Angle-based smoothing. Zhou and Shimada [125] proposed a smoothing method which aims to improve the geometric quality of a mesh in a computationally easy way. Their method has been shown to provide better results than Laplacian smoothing in quality and to avoid creating inverted elements. In this method, angles in a local neighborhood is approximately distributed in equal to achieve better quality elements.

Optimization-based smoothing. Optimization-based smoothing methods are also popular and widely studied, but usually expensive and might lead to a local optimum instead of the global optimum [35, 60, 120]. Therefore, in order to benefit from mesh

optimization techniques in a faster way, combination of different smoothing strategies are also studied [59, 91].

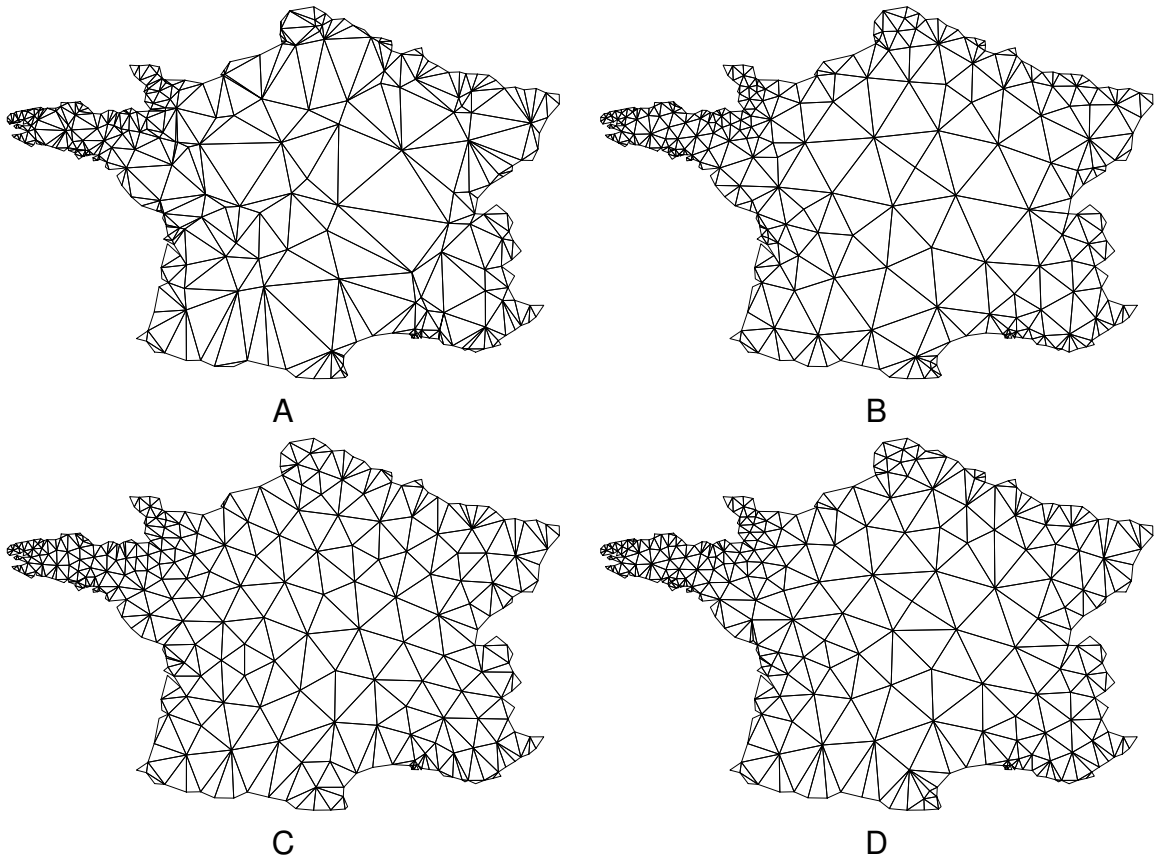


Figure 2-14. Applying different smoothing methods on a given input mesh. A) Initial mesh. B) Laplacian smoothing output. C) CVT-based smoothing output. D) Angle-based smoothing output.

Figure 2-14 demonstrates different smoothing methods, where each technique provides a better quality mesh than the input triangulation. Smoothing is considered as a separate operation which is outside of the mesh (triangulation) generation process. However, few researchers also studied integration of relocation and refinement in different ways [32, 53, 117].

2.3 Steiner Quality Triangulations with Angle Constraints

The preferred shape of a triangle is not exactly the same for all applications. Theoretical and experimental analysis (of numerical methods that are used in conjunction

with triangulations) suggests that triangles with no large angles and/or small angles serve well in most applications [6, 105]. In general, the better the shape of the triangles, the smaller the interpolation and approximation errors are in their use. A triangle is usually considered to have a good shape if its smallest angle is bounded from below (by a user-specified constraint angle α), which implies an upper bound on its largest angle as well ($\pi - 2\alpha$). There are also applications which requires non-trivially small large angles, where algorithms focusing on large angles while generating triangulations are needed. Unless new vertices are added into the domain, even using Delaunay triangulations, which are optimum in maximizing the smallest angle [43], may result in output that involve bad quality triangles. Hence, quality Steiner triangulations are employed for efficiency and effectiveness in applications. However, it is important to keep the triangulation size small while having the shape of its triangles good, in particular the angles. Here, we generalize the quality Steiner triangulation problem with angle constraints as follows:

Problem 1. $[\alpha, \gamma]$-*Triangulation. Given a two-dimensional domain (a set of points or a planar straight line graph), and minimum and maximum angle constraints* α *and* γ*, compute a Steiner triangulation of the domain such that all angles are in the interval* $[\alpha, \gamma]$.

Following well-known problems can be defined in terms of Problem 1:

- No small angle triangulation problem is a special case of Problem 1, where $\gamma = \pi - 2\alpha$.
- No large angle triangulation problem is a special case of Problem 1, where $\alpha = 0°$.
- Non-obtuse triangulation problem is a special case of Problem 1, where $\gamma = \pi/2$.
- Acute triangulation problem is a special case of Problem 1, where $\gamma = \pi/2 - \epsilon$ (ϵ is a small constant).

Combining the objectives of these well-studied problems gives us a more complicated problem than either of them. For instance, it is clear that Delaunay triangulation (i.e.,

max-min angle triangulation) of the refined point set should be used for solving the no small angle problem and min-max angle triangulation should be used for solving the no large angle problem. However, it is not clear which triangulation should be employed in order to generate the output triangulation for Problem 1.

2.3.1 No Small Angle Triangulations

2.3.1.1 Delaunay refinement

Since Delaunay triangulations are known to maximize the smallest angle, many employed them to generate no small angle triangulations [11, 26, 28, 61, 87, 99]. The Delaunay refinement method involves first computing an initial Delaunay triangulation of the input domain and then iteratively adding Steiner points to improve the quality of the triangulation. Various types of Steiner points are studied in the literature, which we review below.

Circumcenters. A natural choice for a Steiner point location is circumcenter of bad triangles (as observed by many [28, 99, 104]). See Figure 2-15 (A). Insertion of its circumcenter surely removes a bad triangle from the Delaunay triangulation (thanks to the empty circle property of Delaunay triangles). Moreover, circumcenters of triangles stand reasonably far from the existing vertices, a property which serves well for proving upper bounds on the output size of Delaunay refinement algorithms [99].

Sinks. Edelsbrunner and Guoy [45] suggested inserting sinks of bad triangles, which are circumcenters of acute triangles. For each bad triangle, an iterative walk in the triangulation, each time crossing the edge opposite to the unique obtuse angle, leads to its sink. See Figure 2-15 (B). Sinks are at the local maxima of the local feature size function [45, 99]. Note that the sink of a bad triangle can be quite far from the triangle itself. Hence, a bad triangle may remain in the triangulation even after its sink is inserted. However, it is shown that iterative insertion of sinks eventually removes all bad triangles in the triangulation. Experiments [45] suggests that the sink insertion

scheme performs similarly to circumcenter insertion. However, we are not aware of any triangulation software that use sinks as the Steiner point insertion strategy.

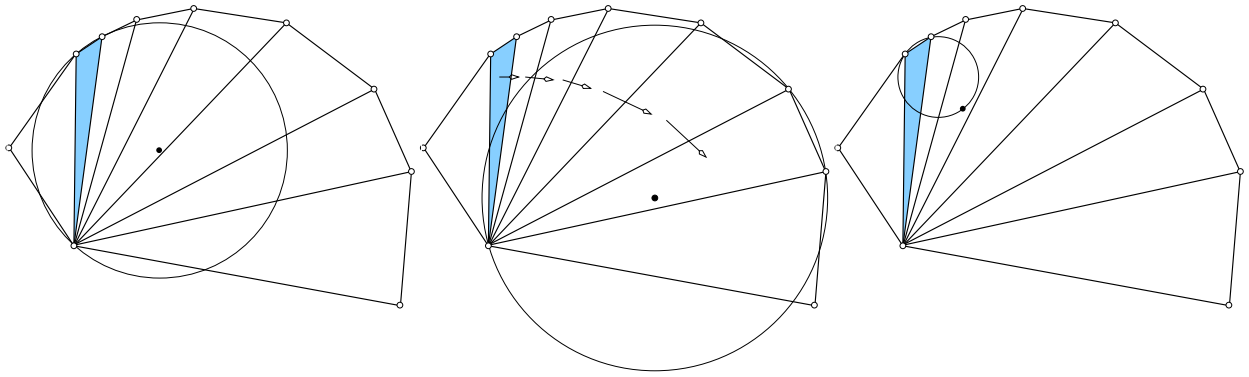


Figure 2-15. Types of Steiner points. A) Circumcenter. B) Sink. C) Off-center.

Off-centers. Üngör introduced another type of Steiner point, called off-center, as an alternative to circumcenters and sinks [118]. Off-centers and circumcenters differ for "very" bad triangles (those with smallest angle at most $\alpha/2$) and are the same for triangles that are almost good (those with the smallest angle between $\alpha/2$ and α). In the former case, the off-center of a triangle with the shortest edge pq is a point o on the bisector of pq furthest from p (or q) such that the angle $\angle poq$ is α. See Figure 2-15 (C). In practice, the off-center insertion algorithm computes significantly smaller triangulations [118]. Off-centers are also numerically more stable than circumcenters and facilitate more robust software. The idea of using off-centers also led to the design of the first time-optimal Delaunay refinement algorithm [68].

Recent years witnessed an inflation in the development and use of Delaunay refinement algorithms [32, 45, 68, 69, 84, 85, 103, 104, 118]. Some of these research studies mainly concentrate on the theoretical bounds [68, 69, 84], while others emphasize the benefits on the practical side [103, 104, 118]. Miller et al. proposed a time-efficient Delaunay refinement algorithm [69, 84]. However, their algorithm currently lacks experimental support to indicate its relevance in practice. None of these

algorithms, however, address the termination problem for large constraint angles which is a primary challenge in order to improve the smallest angle bound.

2.3.1.2 Other refinement algorithms

The quadtree refinement method provides similar bounds (on the smallest angle and the output size) as Delaunay refinement algorithms [13]. However, in practice quadtree refined meshes are significantly larger than Delaunay meshes. Moreover, the quadtree implementations lack the flexibility for the user to specify a constraint angle. Instead, a fixed angle (exact value of which depends on the type of input) guarantee is provided. For polygonal domains, it is proven that the quadtree method can compute output triangulations whose angles are at least 18.4°. In practice, it is common to observe angles that are around 20° in the quadtree refined triangulations.

There are other refinement strategies which are used in conjunction with Delaunay triangulations. For instance, the longest-edge propagation path methods use the midpoint of the longest edge of certain triangles as the Steiner point [98]. Theoretical bounds for these methods tend to be relatively weaker than that of the Delaunay refinement methods. Moreover, these methods also suffer from the termination problem in practice for constraint angles even smaller than 34°.

2.3.2 No Large Angle Triangulations

There has been extensive research on different versions of Problem 1. For instance, there are algorithms for computing a non-obtuse triangulation of simple polygons (usually with holes) [9, 12, 13, 15, 49]. Since most of these algorithms use axis-aligned structures, such as quadtrees, rectangular grids or vertical/horizontal lines during their construction process, configuration of the output triangulation and its size are negatively affected [9, 12, 13]. Distinctly, Bern et al. [15] and Eppstein [49] utilize non-overlapping disks for generating non-obtuse triangulations. However, the angles in the output can still be arbitrarily small.

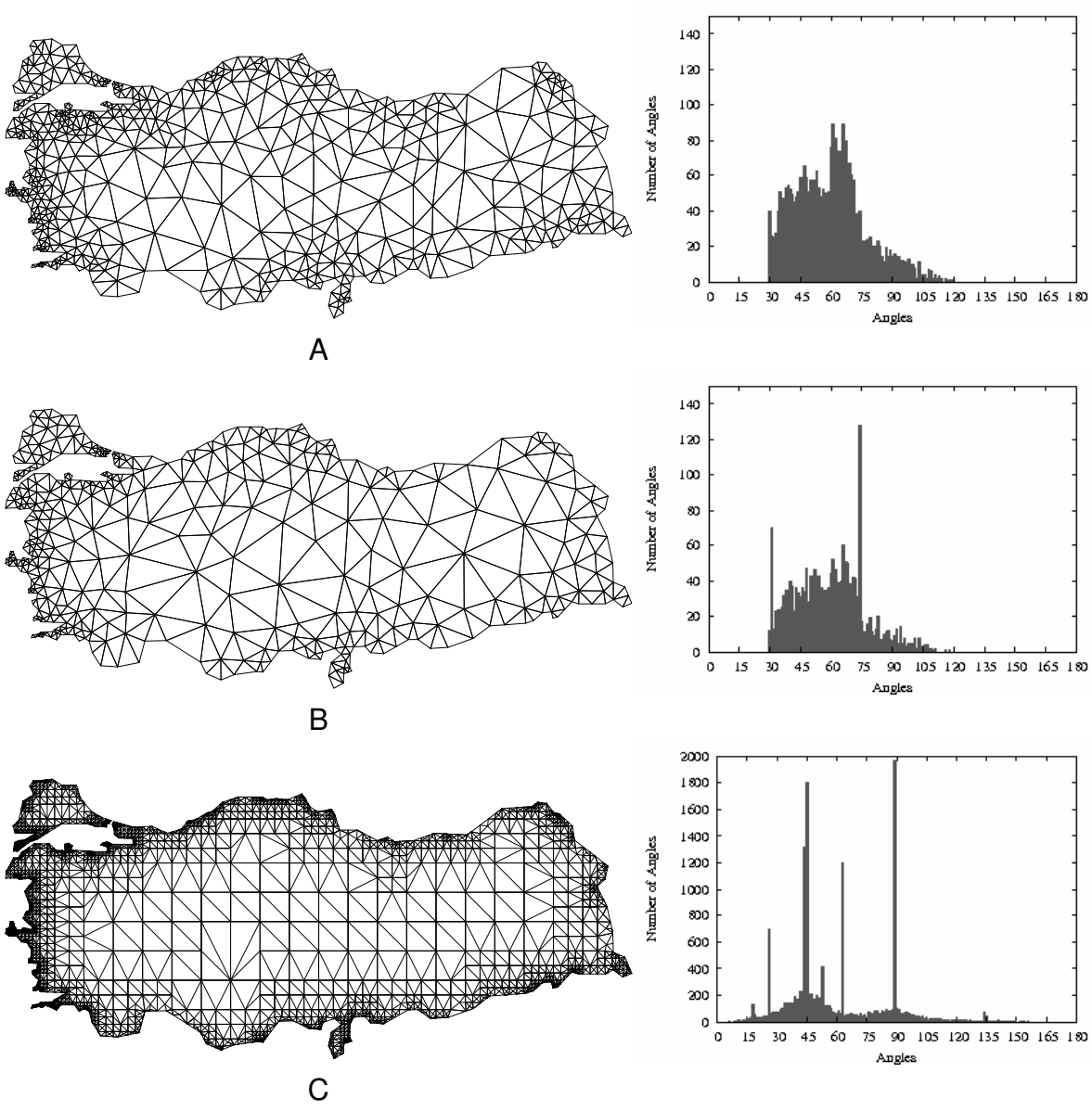


Figure 2-16. Output of different types of refinement algorithms and their angle histograms. A) Circumcenter insertion, $[\alpha, \gamma] = [30°, 119°]$, 613 points. B) Off-center insertion, $[\alpha, \gamma] = [30°, 118°]$, 467 points. C) Quadtree refinement, $[\alpha, \gamma] = [6°, 156°]$, 2893 points.

For planar straight line graphs, Mitchell [90] and Tan [114] proposed algorithms based on propagating paths of Steiner points to eliminate bad quality triangles. These algorithms give $157.5°$ and $132°$ bounds on large angles, respectively. Miller et al. [86] recently introduced a simpler algorithm which uses Delaunay refinement. However, their upper bound of $170°$ is worse than the bounds provided by the previous algorithms and

this bound is not strong enough to solve the acute and the non-obtuse triangulation problems. While the method of Miller et al. is size competitive, the output size is not optimal.

Algorithmic and existence results for acute triangulations are also known when the input is a point set or a low-complexity polygon such as a square, a quadrilateral or a pentagon [13, 22, 79, 80]. In their seminal paper, Bern et al. [13] present a variant of their quadtree refinement algorithm in order to generate $[\alpha = 36°, \gamma = 80°]$-triangulations. However, the input for their method is limited to point sets. Moreover, output size of this algorithm tends to be too large because it requires extra level of refinement on the quadtree.

Also, some studies considered no small angle condition with non-obtuseness. Baker et al. [7] employed a regular grid to construct non-obtuse triangulations whose smallest angle is as large as $13°$. Melissaratos and Souvaine [82] improved this by integrating strategies of Baker et al. [7] and Bern et al. [13] and providing output size guarantee, in theory. However, these algorithm tend to be complicated and as they rely on grid structure, output size for these algorithms are not promising to be competitively small in practice. Indeed, we are not aware of any software based on these methods.

2.3.3 Smoothing

Mesh smoothing schemes come as an alternative to the mesh refinement algorithms discussed above. They aim to improve the element shape and sizes as well, however only relocation process is allowed. Some approaches are also apply edge flipping operations, whereas others keep the initial connection information throughout the process.

Geometry-based methods. Although these smoothing methods are shown to improve angle qualities in general, they usually prone to generate invalid elements and has limited strength. Among these methods, angle-based smoothing [125] specifically focuses on angles, while its basic idea relies on improving the angle distributions. In

particular, for complex geometric domains popular geometry-based smoothing methods fail to represent the domain boundary and introduce triangulations with small and/or large angles. A detailed comparative study on several smoothing methods can be found in [55]. In this study [55], inconsistent behaviors of smoothing methods have been observed, especially when the input domain contains complex features. That is, output examples of some smoothing methods with significant decrease in small and large angle bounds are illustrated (see also Figure 2-17).

Optimization-based methods. These methods take an already refined mesh as input and iteratively relocate its vertices in order to optimize a quality criterion [3, 35]. Optimization-based smoothing methods are generally less efficient than the refinement methods as it takes several iterations for the optimization to converge. Combination with simple geometry-based approaches has been also considered [59, 91]. These algorithms mostly focus on minimizing the smallest angle rather than the largest angle, since the latter is known to be harder than the former due to the non-convex nature of the problem [3]. However, a recent well-centered triangulation (WCT) method of VanderZee et al. [120] contains an optimization function which aims to compute acute triangulations (see Figure 2-17). Experiments indicate that WCT method generally improves the largest angle but does not necessarily produce acute angle triangulations and also sometimes deteriorates the smallest angle. Note that, since smoothing methods work on existing meshes, the improvement on the angle quality strongly related to the quality of the initial triangulation and expected to be limited.

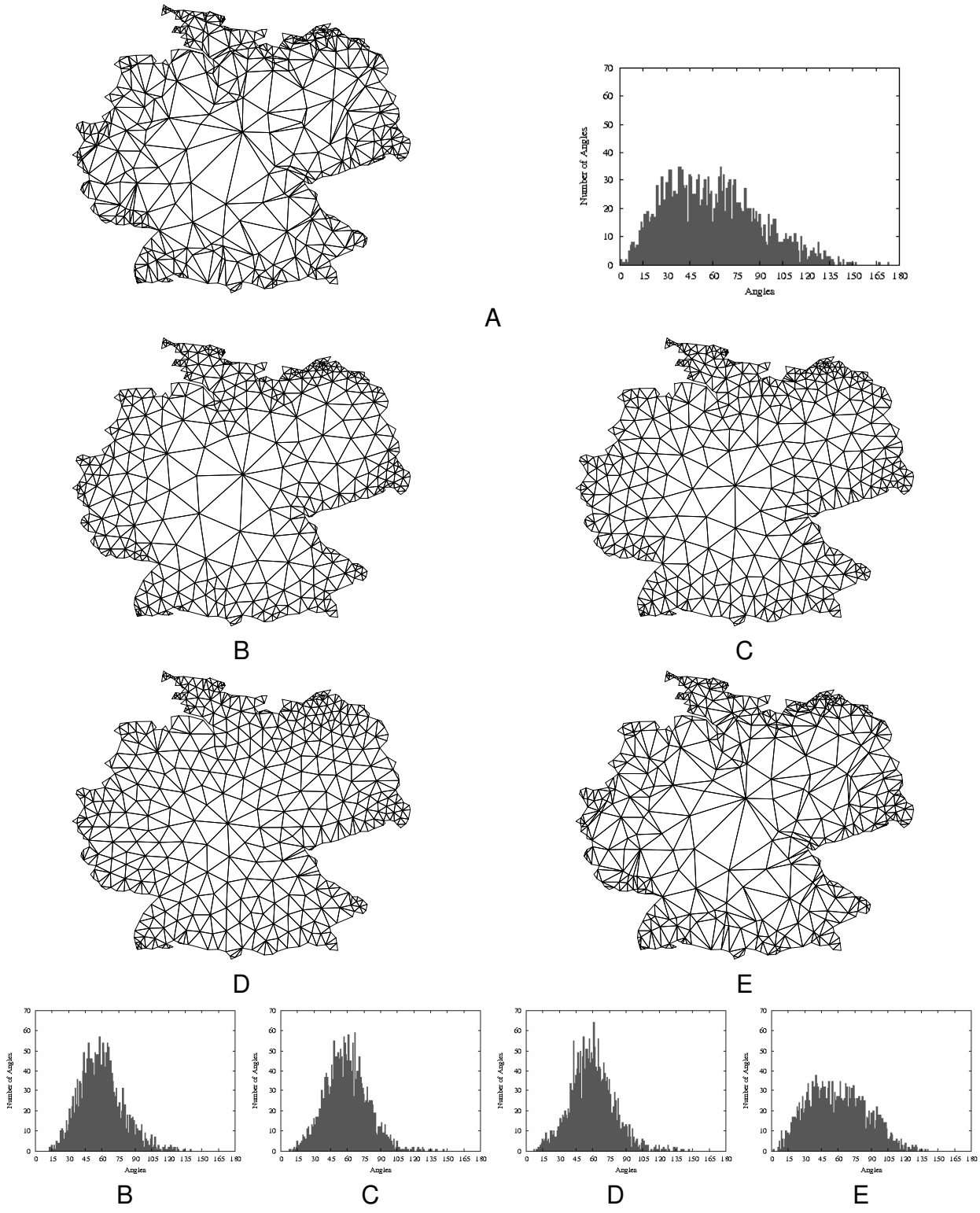


Figure 2-17. Output of different types of smoothing algorithms and their angle histograms. A) Initial triangulation, $[\alpha, \gamma] = [0.65°, 173°]$. B) Laplacian smoothing, $[\alpha, \gamma] = [13.3°, 140°]$. C) Angle-based smoothing, $[\alpha, \gamma] = [8.3°, 147°]$. D) CVT-based smoothing, $[\alpha, \gamma] = [7.7°, 147°]$. D) WCT smoothing, $[\alpha, \gamma] = [2.65°, 141°]$.

CHAPTER 3
TRIANGULATIONS WITH LOCALLY OPTIMAL STEINER POINTS

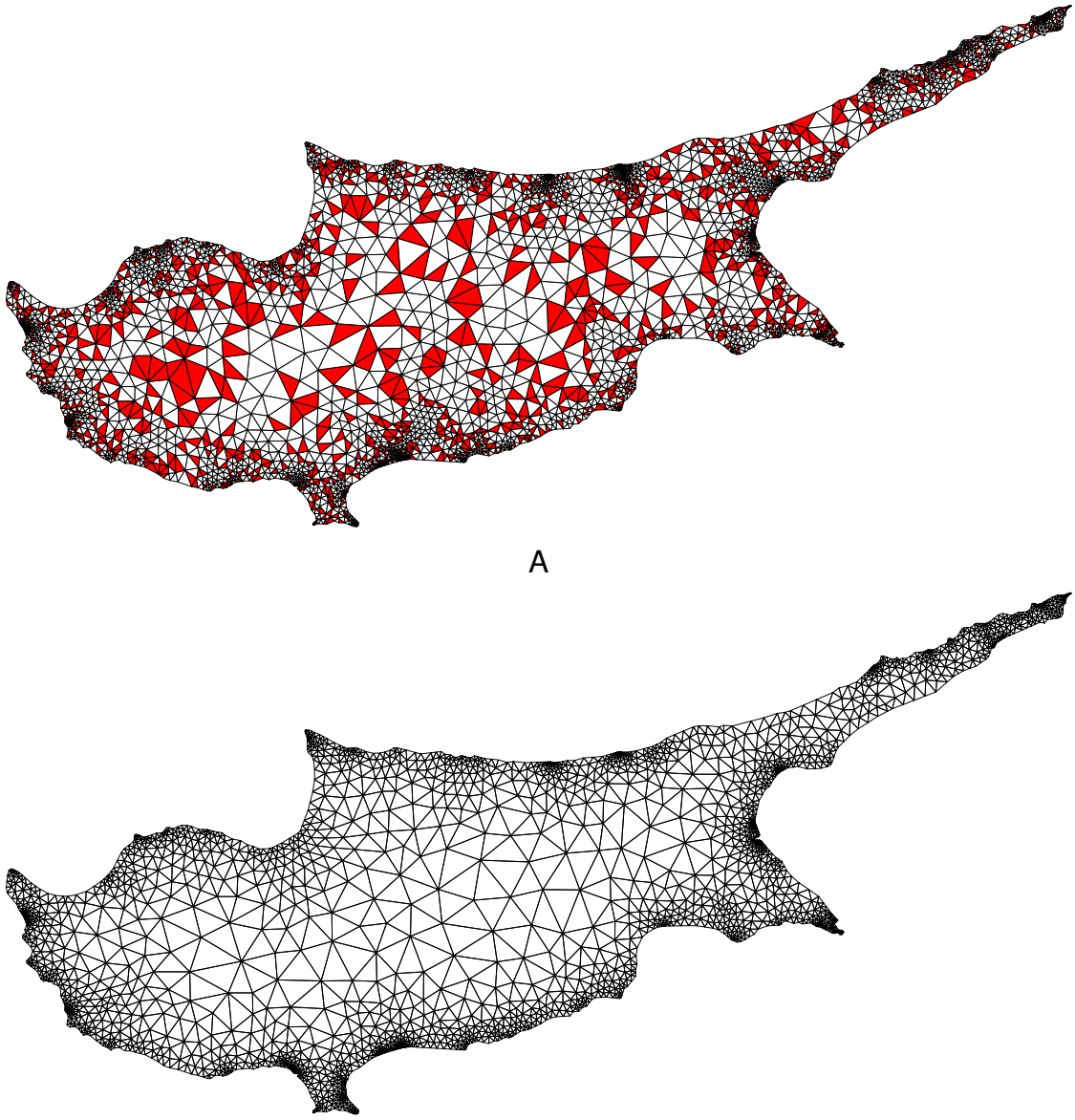
3.1 Introduction

Several approximation algorithms have been suggested for solving the quality Steiner triangulation problem [13, 99]. Approximation guarantee for a specific Steiner triangulation algorithm is usually expressed as follows: Given a two-dimensional domain and a constraint angle $\alpha \leq \beta$, the algorithm computes a triangulation of the input domain whose size is within a constant factor of the optimal such that all the angles of the triangulation are at least α. The angle β should be thought of as the theoretical limit of an algorithm. The exact value of the β depends on the approach/algorithm as well as the input type. For instance, the Delaunay refinement algorithm of Ruppert [99] is proven to compute good triangulations for $\alpha \leq \beta = 30°$ on point sets and for $\alpha \leq \beta = 20.7°$ on planar straight line graphs.

In practice, the lack of theoretical guarantee for a Steiner triangulation algorithm, when $\alpha > \beta$, is observed as a never-ending refinement process. (See Figure 3-1 (A).) Steiner points are iteratively inserted until the computer exceeds its numerical precision capacity or its memory. Shewchuk's experimental study revealed that the Delaunay refinement algorithm in practice works better than its theoretical guarantee:

> Ruppert [99] proves that this procedure halts for angle constraint of up to $20.7°$. In practice, the algorithm generally halts with an angle constraint of $33.8°$, but often fails to terminate given an angle constraint of $33.9°$. It would be interesting to discover why the cut-off falls there. [101]

There have been attempts to explain this cut-off [85]. Here, we take a different strategy, and, rather than explaining this limitation, we remove it. We present a new Delaunay refinement algorithm which sets a new cut-off. This cut-off angle will be referred to as ζ, representing the empirical limit for Delaunay refinement algorithms. Our refinement algorithm terminates with good triangulations for constraint angles up to $42°$, i.e., $\alpha \leq \zeta = 42°$. Figure 3-1 illustrates the output of our algorithm in contrast with the behavior of previous Delaunay refinement algorithms.



A

B

Figure 3-1. The original Delaunay refinement algorithm does not terminate for constraint angles larger than 34°. Our algorithm computes quality triangulations for constraint angles up to 42°. For example, when the constraint angle is 40°, the original Delaunay refinement chokes with bad (shaded/red) triangles (A), whereas our algorithm computes a nicely graded triangulation where all angles are larger than or equal to the constraint angle (B).

3.1.1 Our Idea and Contribution

Existing refinement algorithms, including the circumcenter and the off-center insertion variants of the Delaunay refinement, suffer from a so-called termination problem for large values of α. In this chapter, we explore ideas to alleviate this problem.

We propose two simple algorithms, where the second is an extension of the first. The first algorithm suggests to locate Steiner points inside a particular neighborhood of bad triangles that are furthest away from existing vertices. One might wonder, Are such points not circumcenters (at least for the almost good triangles)? It turns out the answer is "No!" for most of the bad triangles in a typical Delaunay triangulation. In Section 3.2, we classify this Steiner point description into four different types. Our second algorithm incorporates a simple relocation scheme which becomes effective especially for large α values. Before attempting to insert a Steiner point for a bad triangle, we simply try fixing the bad triangle by relocating one of its vertices (if that vertex was inserted as a Steiner point).

These ideas lead to the following as the main contributions of this Chapter, which we elaborate more via experiments:

- Unification of Steiner points.
- Effective and efficient integration of smoothing and refinement.
- Quality improvement on practical small angle bounds.
- Sparse well-spaced point sets.
- Faster triangulation software.

3.1.2 Motivation

The primary motivation for studying quality triangulations still applies here. That is, we would like to compute better quality, smaller size triangulations as fast as possible. In this section, we are more specific about the motivation of this chapter and highlight

two important issues that continue to be challenges at the current state of quality triangulation research.

3.1.2.1 Termination problem

Termination guarantee of the Delaunay refinement algorithms relies on a packing argument. This argument applies for small values of α (i.e., $\alpha \leq 30°$), where circumcenter insertion does not gradually decrease the shortest pairwise vertex distance. For $\alpha > 30°$, however, the iterative refinement process could introduce smaller and smaller features and may not terminate (see Figure 3-2). In practice, such a phenomenon starts occurring for $\alpha > 34°$, regardless of the type and distribution of input data. See Figures 3-3 and 3-4. Coincidentally, all known previous versions of the Delaunay refinement [28, 68, 84, 99, 102, 118] have the termination problem around $34°$, as further explained in the next section.

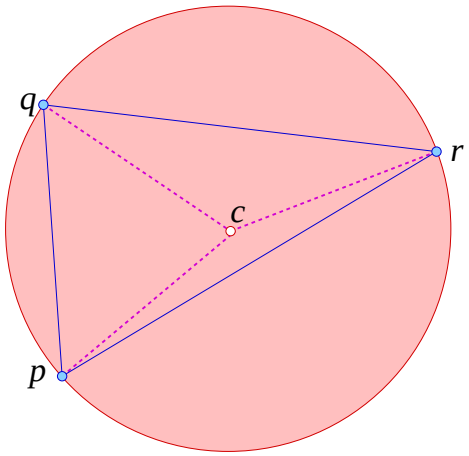


Figure 3-2. When $\alpha > 30°$, circumcenter insertion introduces shorter pairwise distances (among vertices of triangulation) than existing ones, i.e., $|pc| = |qc| = |rc| < |pq|$.

Different versions of available software. Any refinement algorithm that makes significant use of circumcenters is expected to suffer from the termination problem for constraint angles not much larger than $30°$. The majority of the points inserted by most Delaunay refinement algorithms are circumcenters [28, 45, 99, 104]. While the circumcenters are less frequently used by the off-center insertion algorithm [118], their

percentage is significant enough that the practical performance bound ζ remains roughly the same. Figure 3-3 illustrates how both Triangle 1.4, which is an implementation of the circumcenter insertion, and Triangle 1.6, which is an implementation of the off-center insertion, suffer from the termination problem. It is important to observe the characteristic difference between these two methods when they get caught in the never-ending refinement process. This difference is due to the order in which the bad triangles are handled. The off-center insertion algorithm works better when the bad triangles with the shortest edges are handled first [68]. On the other hand, the circumcenter insertion algorithm works better when the bad triangles with the smallest angles are handled first. When they terminate, the off-center insertion performs better than the circumcenter insertion; that is, it outputs meshes with fewer triangles [101]. However, when interrupted in the never-ending refinement process, the evolving mesh of the off-center refinement looks worse than that of the circumcenter refinement. This can be fixed by changing the bad triangle handling order to smallest length first.

With or without (small input angle) constraints. Prior to our work, the termination problem was addressed in the context of small input angles [85, 94, 103]. It is important to note that in general the termination problem exists regardless of the simplicity of the input domain. See Figure 3-4, where the input consists of a pair of points enclosed by a large hexagon. In general, for any input domain, it is common to observe overrefinement occurring in regions far away from the input features whenever α is large. In this study, we do not limit ourselves to certain types of constraints and rather address the termination problem in general. Our analysis in Section 3.3 focuses on point sets. However, this work can be easily coupled with the previous work [85, 104, 118] that considered planar straight line graphs. Our experiments, on data sets that are planar straight line graphs with small input angles, verify this statement.

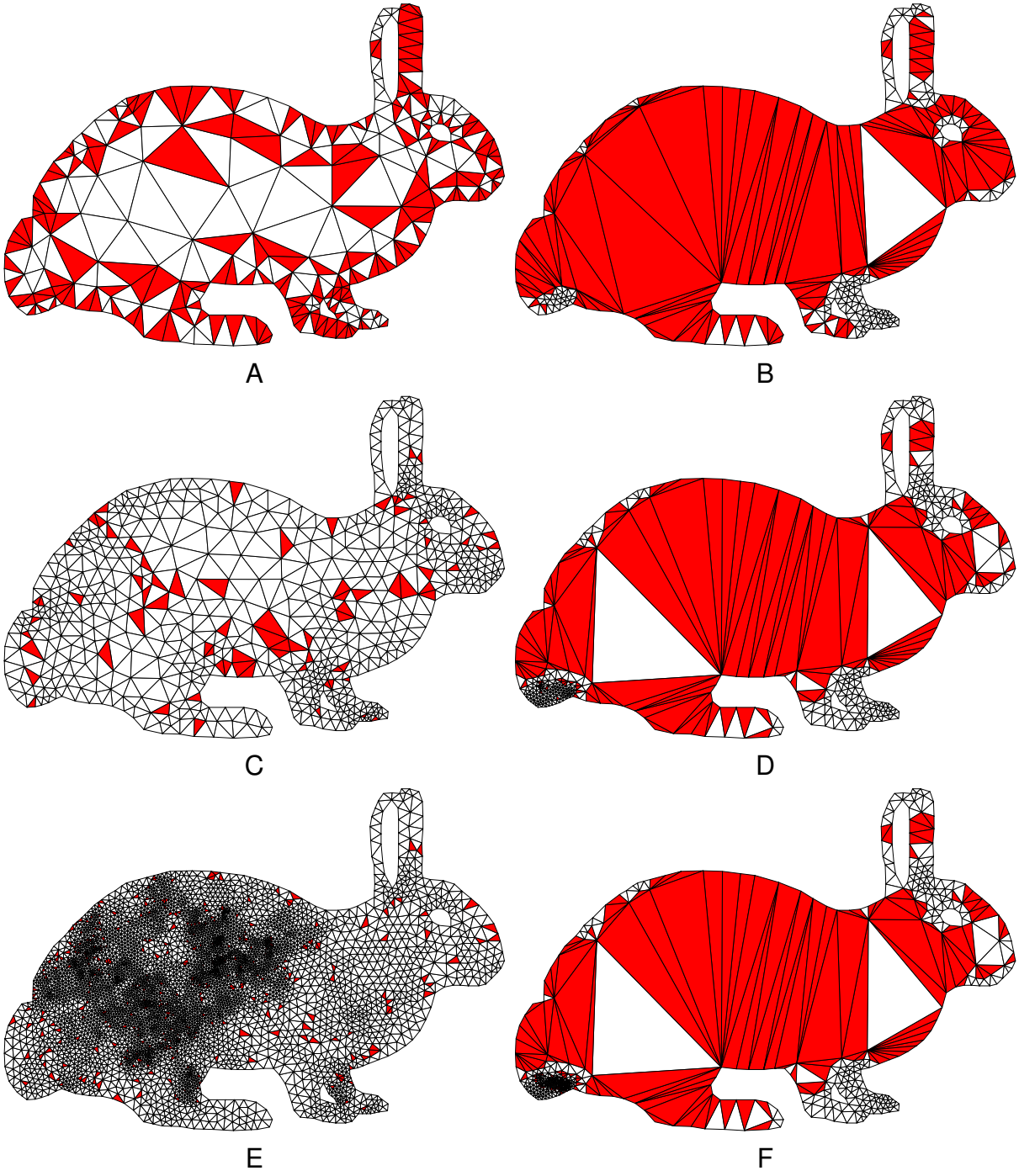


Figure 3-3. Termination problem shown for the constraint angle $\alpha = 36^\circ$ on the rabbit data set. Bad triangles are shaded (red). Left A, C, and E: evolving triangulation using the circumcenter insertion (Triangle 1.4) using 100, 500 and 5000 Steiner points, respectively. Right B, D, and F: evolving triangulation using the off-center insertion (Triangle 1.6) using 100, 250 and 500 Steiner points, respectively. Neither of the two algorithms terminates. So, for this illustration, we interrupted their execution after the insertion of S Steiner points.

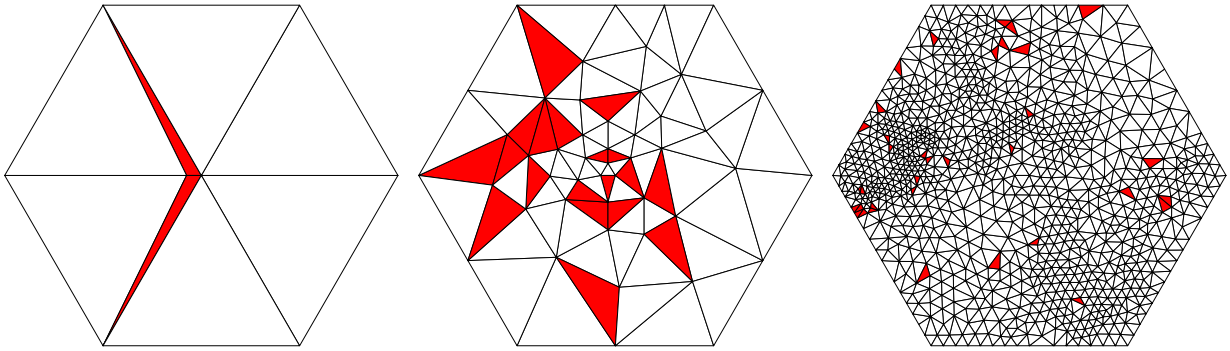


Figure 3-4. Termination problem is shown for $\alpha = 35°$ on a simple data set: a pair of points at unit distance from each other inside a hexagon of side length 12 units. Bad triangles are shaded (red). A) Delaunay triangulation of the input which includes two bad triangles. B and C) Evolving triangulation using the circumcenter insertion (Triangle 1.4). The refinement process does not terminate, and hence the execution is interrupted after the insertion of 0, 50 and 1000 Steiner points.

3.1.2.2 Mesh size

The output size is crucial for the efficiency of the methods using these meshes in various applications. Also, the smaller the number of Steiner points, the faster the refinement algorithm will be (if the Steiner point computation time is kept the same). Simpson [109] recently showed empirical evidence that standard Delaunay refined meshes (through circumcenter insertion) are roughly twice the size of the optimal meshes for an application he calls function approximation. Our results here complement this study as we get a size improvement of around a factor of two compared to the previous Delaunay refinement algorithms.

3.2 Locally Optimal Steiner Points

Definition 12. *Given an ordered pair of points (p, q) in the input domain, consider the circle that goes through p and q, of radius $|pq|/(2\sin\alpha)$, whose center is on the right side of the directed edge pq. The disk bounded by this circle is called the petal of pq, denoted by $petal(pq)$.*

Note that every edge has two petals (of interest), unless both of its end points are on the boundary. In a triangulation, petals (of interest) of every edge pq must include another vertex. Otherwise, there would be a bad triangle in the triangulation incident to p and q. Har-Peled and Üngör [68] proved that the triangulation of a point set is good if and only if the petal of every pair of points contains another point. In the following algorithm, we suggest one to pick a Steiner point inside empty petals furthest away from all existing vertices; this is the concept of locally optimal point location.

Algorithm 2.

Compute the Delaunay triangulation of the input
while $\exists$ a bad triangle pqr with shortest edge pq
  insert a point $x \in petal(pq)$ of its shortest edge pq
    that is furthest from all existing vertices

This point is either a Voronoi vertex (but not necessarily the circumcenter of pqr) or on a Voronoi edge (see Lemma 1). We classify these points into four different types in order to relate them with the existing refinement strategies. This classification also helps us to present a theoretical and experimental assessment of our method. Let pqr be a bad triangle whose shortest edge is pq. Then, the point inside the petal of pq furthest away from all existing vertices is one of the following types (see Figure 3-5).

- Type I. The intersection of the Voronoi dual of pq and the boundary of the petal of pq. This case happens when the circumcenter of pqr is outside the petal of pq. This type is nothing but the off-centers described in [118].
- Type II. A point on a Voronoi edge other than the dual of the shortest edge pq. This is a new type of Steiner point for Delaunay refinement and should be seen as an extension of the off-center idea to the triangles that are almost good (circumcenter of pqr is inside the petal of pq).
- Type III. The circumcenter of a nearby triangle to pqr. Such a triangle must be acute. So, this type is a sink circumcenter. However, it is not necessarily the sink of the considered bad triangle pqr (as is the case in Figure 3-5 (C)). Hence, our use of sinks is somewhat different than the original sink insertion algorithm of Edelsbrunner and Guoy [45].
- Type IV. The circumcenter of the bad triangle pqr.

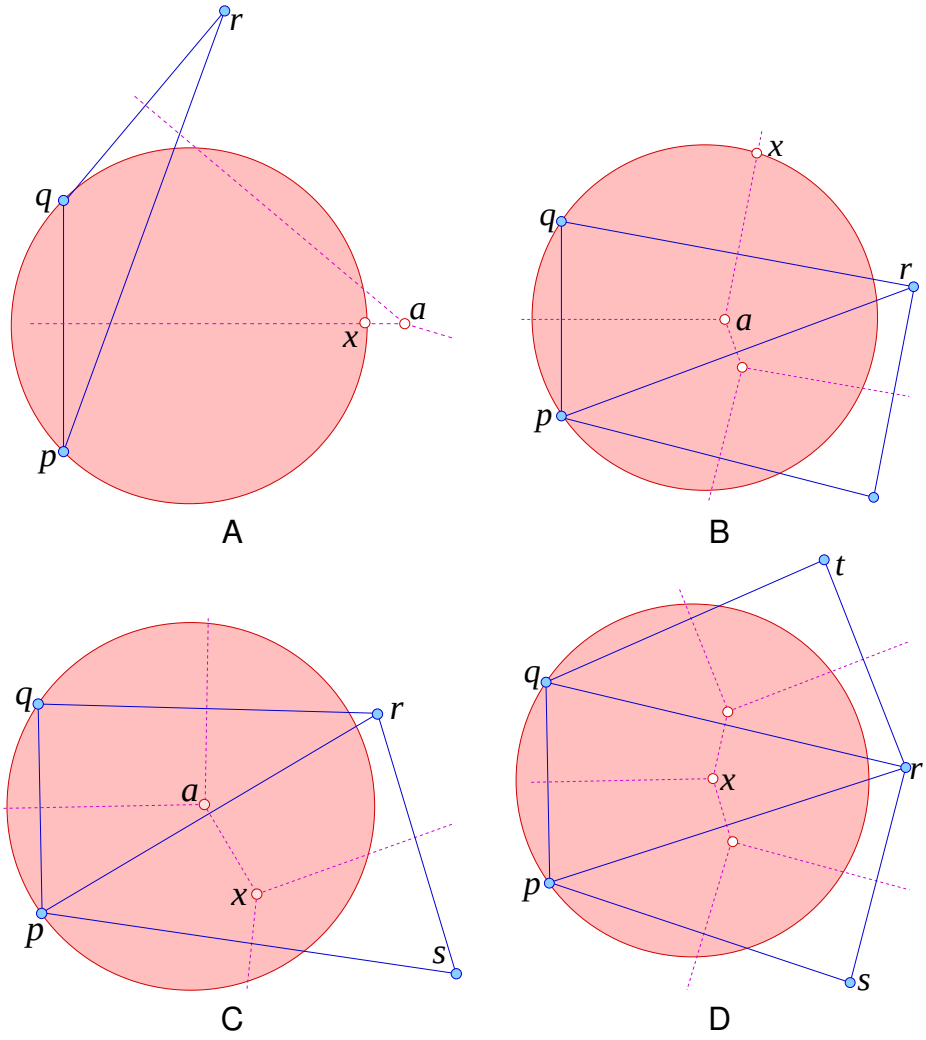


Figure 3-5. Find a point x inside the (shaded disk) petal of pq that is furthest away from all existing vertices. Such a point can be on a Voronoi edge (A and B) or a Voronoi vertex (C and D). A) Type I: on the dual of pq. B) Type II: on a Voronoi edge other than the dual of pq. C) Type III: a circumcenter other than that of pqr. D) Type IV: the circumcenter of pqr.

Note that the furthest point in an arbitrary region from all existing vertices is not necessarily on the Voronoi skeleton. A straightforward example of this is a closed region which does not contain any Voronoi edge or Voronoi vertex. A locally optimal point limited to such a region is clearly on its boundary. On the other hand, Figure 3-6 (A) illustrates a more interesting example where the region of interest (shaded) intersects the Voronoi skeleton. Note that the region in this example is convex and also contains two vertices on its boundary, two of the properties of a petal. The locally optimal Steiner

point in the shaded region, marked as x, is not on the Voronoi skeleton. However, inside a petal the furthest point from all existing vertices has to lie on the Voronoi skeleton. This nontrivial result is given in Lemma 1.

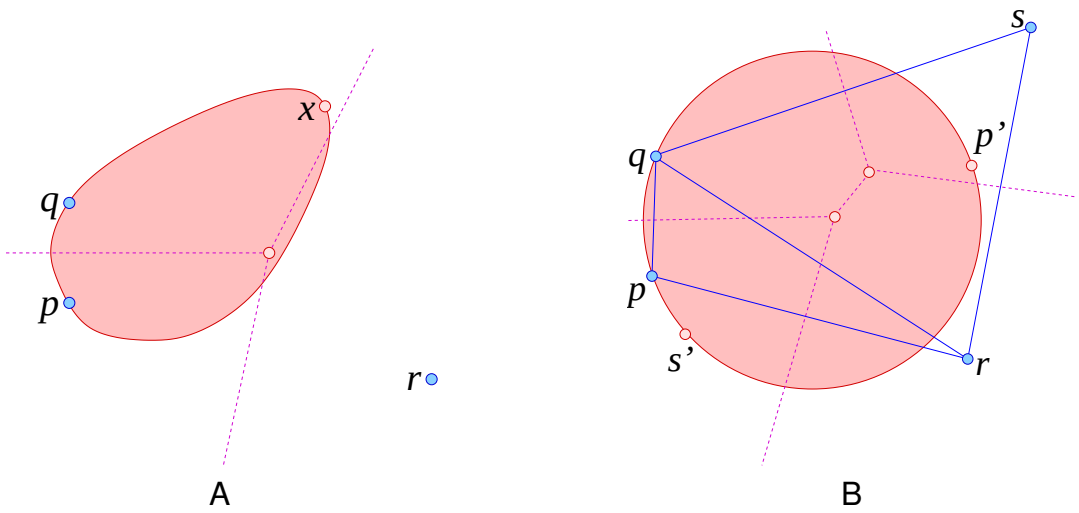


Figure 3-6. A) Locally optimal Steiner point limited to a convex region. B) Illustration of the Lemma 1 proof.

Lemma 1. *For an evolving point set, a locally optimal (furthest from all existing vertices) Steiner point in a petal region is either a Voronoi vertex or on a Voronoi edge.*

Proof. Consider a bad triangle pqr with the shortest edge pq. The $petal(pq)$ is divided into regions bounded by arcs (belong to $\partial petal(pq)$) and Voronoi edges. Let x be the locally optimal Steiner point limited to $petal(pq)$. First observe that x is on $\partial petal(pq)$, the Voronoi skeleton, or both. Otherwise, there is always a point around x that is further from all vertices than x is, which contradicts the optimality of x. For the sake of contradiction, now suppose that x is on $\partial petal(pq)$ but not on the Voronoi skeleton. Let a be the site whose Voronoi region contains x, i.e., $x \in \text{Voronoi}(a)$. We complete the proof by considering two cases (i) $a \in \{p, q\}$ and (ii) $a \notin \{p, q\}$.

(i) Without loss of generality, let $a = p$, i.e., $x \in \text{Voronoi}(p)$. Let p' be the furthest point from p in $petal(pq)$; that is, p and p' are antipolar. (See Figure 3-6 (B).) Since $|p'q| < |p'p|$, $p' \notin \text{Voronoi}(p)$, and hence $p' \neq x$. Also, note that for any point other than p' on $\partial petal(pq)$ it is possible to find another point further from p by walking on $\partial petal(pq)$

towards p'. This in turn leads to a contradiction as x cannot be optimal unless it is both on $\partial petal(pq)$ and on the Voronoi skeleton.

(ii) Let $x \in \text{Voronoi}(s)$, where $s \notin \{p, q\}$. The argument is similar to the case (i). Let s' be the furthest point from s in $petal(pq)$ (see Figure 3-6 (B)). Observe that either p or q is closer to s' than s is. So, $s' \notin \text{Voronoi}(s)$, and hence $s' \neq x$. Also note that, for any point other than s' on $\partial petal(pq)$, it is possible to find another point further from s by walking on $\partial petal(pq)$ towards s'. This in turn leads to a contradiction as x cannot be optimal unless it is both on $\partial petal(pq)$ and on the Voronoi skeleton. □

We compute the location (and also the type) of the Steiner point simply by doing a local search (breadth-first search) on the Voronoi graph. The angles of the visited triangles provide guidance in this search. For instance, if the smallest angle of the bad triangle is less than $\alpha/2$, then we know for sure that the locally optimal Steiner point is of Type I (an off-center). In general, it is better to start the search visiting the triangle opposite to the largest angle of the bad triangle. We elaborate more on the implementation details of the algorithm in Section 3.5.

3.3 Analysis

The analysis of the Delaunay refinement algorithms relies on lower bounds on the local feature size of the Steiner points. It is relatively easy to extend the analysis of the standard refinement techniques for our algorithms, since the local feature size of the Steiner points we use is not any smaller than that of the circumcenters. See, for instance, [118] for an analysis of the refinement that uses Type I and Type IV Steiner points. Indeed, one might expect an improvement on the angle bound. Such an improvement, however, seems difficult to prove and is left as an open problem. Here, we provide some theoretical evidence on why our strategy works for large values of constraint angle α. For this purpose, the following lemma should be complemented with the experimental results on the percentage use of each type of Steiner point (presented on Tables 3-2 and 3-3 in Section 3.5).

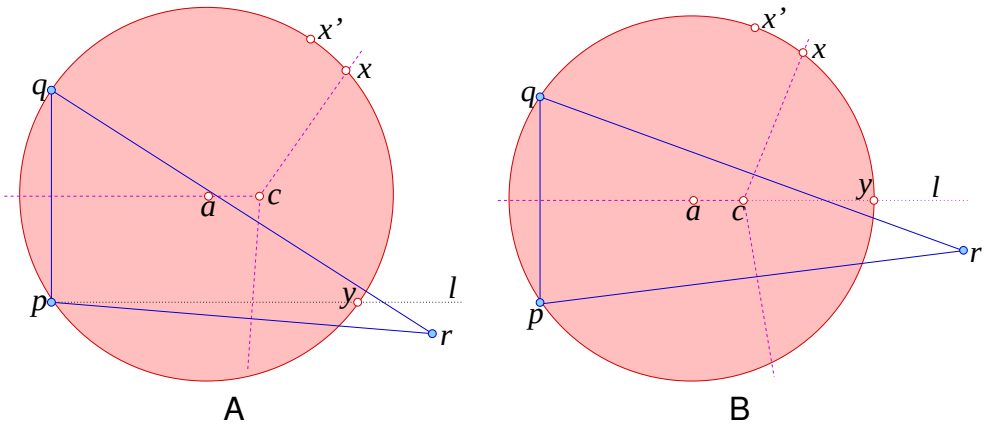


Figure 3-7. A) Type II (A). B) Type II (B).

We further classify the Type II vertices for the sake of the analysis. Two special cases of Type II will be analyzed separately, where the Steiner point x is on the dual of longest edge of the bad triangle. In other words, consider a Steiner point that is on the dual edge of qr of a bad triangle with shortest edge pq, where qr is the longest edge. This Type II Steiner point is called a Type II (A) if $\angle qpr \geq \pi/2$ and a Type II (B) if $\angle pqr \leq \angle qpr < \pi/2$.

Lemma 2. *Given a point set Ω and a desired minimum angle α as input, the Algorithm 2 does not create any feature shorter than the existing ones while inserting a vertex x of*

(a) Type I *Steiner points if* $\alpha \leq \pi/3 = 60°$*;*

(b) Type II (A) *Steiner points if* $\alpha \leq \pi/4 = 45°$*;*

(c) Type II (B) *Steiner points if* $\alpha \leq \pi/5 = 36°$*;*

(d) Type II*, Type* III*, and Type* IV *Steiner points if* $\alpha \leq \pi/6 = 30°$*.*

Proof. Let pqr be a bad triangle with shortest edge pq.

(a) Since x is on the Voronoi edge of pq, its nearest neighbors among the existing vertices are p and q. Observing that $|xp| < |pq|$ if and only if $\angle pxq > \pi/3$ completes this part of the proof.

(b) Assume, without loss of generality, that x is on the Voronoi dual of qr. So, $\angle qpr$ is a non-acute angle. Let l be the line orthogonal to pq which goes through p

(Figure 3-7 (A)). Let y be the other (than p) intersection of line l and $\partial petal(pq)$. Without loss of generality, assume that $petal(pq)$ is unit disk. Then, the length of the arc pq is 2α (in radians). Let x' be the midpoint of the arc yq. This means the length of the arc $x'q$ is $\pi/2$. Hence, $\alpha \leq \pi/4$ if and only if $pq \leq x'q$. Note that there is no point below the line l and outside the $petal(pq)$ that is closer to x' than q. So, $|x'q| \leq |xq|$. Then, we conclude that $|xq| \geq |pq|$ if $\alpha \leq \pi/4$.

(c) This part of the proof is similar to (b). Assume, without loss of generality, that x is on the Voronoi dual of qr. So, $\angle pqr \leq \angle qpr < \pi/2$. Let l be the line orthogonal to pq and going through the midpoint of pq (Figure 3-7 (B)). Let y be the intersection point of line l and $\partial petal(pq)$ that is furthest from pq. Without loss of generality, assume that $petal(pq)$ is unit disk. Then, the length of the arc pq is 2α (in radians) and the length of the arc yq is $\pi - \alpha$. Let x' be the midpoint of the arc yq. This means the length of the arc $x'q$ is $\pi/2 - \alpha/2$. Hence, $\alpha \leq \pi/5$ if and only if $pq \leq x'q$. Note that there is no point below the line l and outside the $petal(pq)$ that is closer to x' than q. So, $|x'q| \leq |xq|$. Then, we conclude that $|xq| \geq |pq|$ if $\alpha \leq \pi/5$.

(d) The proof is straightforward to show. □

Lemma 2 suggests that if we could show that the Algorithm 2 only uses, say, Type I and Type II (A) Steiner points, then it would (provably) compute triangulations with a minimum angle of $45°$. Unfortunately, such a premise does not seem plausible. Our experiments (presented in Section 3.5) show that all four types are employed by Algorithm 2 in different amounts.

The following theorem follows from the above lemma. Its proof is similar to the results in traditional Delaunay refinement [99, 104, 118] and hence omitted here.

Theorem 3.1. *Given a planar straight line graph Ω and a desired minimum angle $\alpha \leq 30°$, Algorithm 2 terminates with a correct output.*

Time complexity. A straightforward running time analysis leads to a quadratic time complexity bound for Algorithm 2 as is the case for most other Delaunay refinement

algorithms. However, this bound can be improved to $O(n \log n + m)$, where n is the input size and m is the output size, by using a technique recently introduced [68]. This technique employs a balanced quadtree as a data structure and takes advantage of the locality of the Steiner points with respect to the shortest edge of bad triangles.

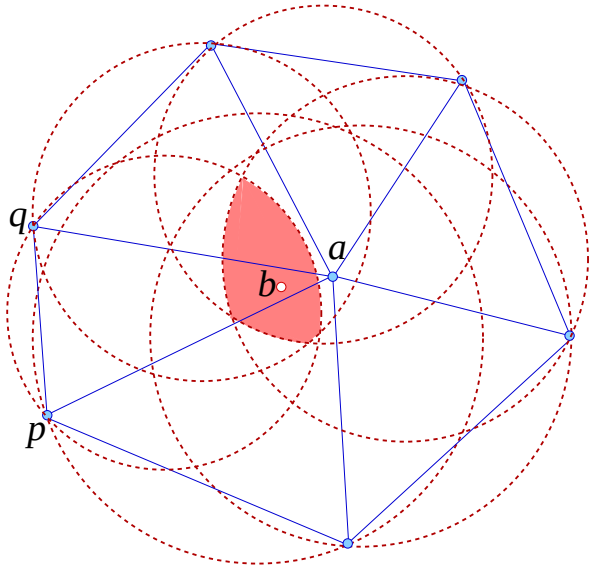


Figure 3-8. Relocating a free vertex of a bad triangle pqr to the intersection of the petals of the link of $a = r$.

3.4 Relocation and Refinement

Based on our analysis (Section 3.3) and experiments with Algorithm 2, we observed that the refinement process introduces shorter edges than existing ones usually when a bad triangle is almost good and also the neighbor triangles are good or almost good. This suggests that it might be easy to fix such bad triangles by a local smoothing strategy (see Figure 3-8). While one might explore various powerful smoothing strategies [3, 35, 60] in these cases, we opt for simplicity and efficiency. We first recall a couple of definitions, some of which are surveyed in [44], and then describe a simple adjustment to our algorithm. The star of a vertex a consists of all triangles that contain a (see Figure 3-8). The link of a then consists of all edges of triangles in the star that are disjoint from a. A vertex is said to be free if it was inserted by the refinement algorithm as a Steiner point. Input vertices are not free and never relocated. For each bad triangle,

we first (one at a time) attempt to relocate its free vertices. If one of its free vertices finds a new location so that all the triangles in its (new) star become good, we perform the relocation. Otherwise, we proceed with a new Steiner point insertion as described in Algorithm 2.

Algorithm 3.

Compute the Delaunay triangulation ($DelTri$) of the input
Let P denote the maintained point set
while $\exists$ a bad triangle pqr in $DelTri(P)$ with shortest edge pq
  relocated := FALSE
  for each free vertex a of pqr
    if $\mathcal{K} = \bigcap_{xy \in link(a)} petal(xy) \neq \emptyset$
      and $\exists b \in \mathcal{K}$ such that all triangles of $star(b)$
        in the $DelTri(P \cup \{b\} - \{a\})$ are good
    then
      delete a; insert b; relocated:=TRUE; **break**;
  endfor
  if relocated == FALSE **then**
    insert a point $x \in petal(pq)$ of the bad triangle's shortest edge pq
      that is furthest from all existing vertices
endwhile

The standard analysis techniques (for proving termination and output size optimality) that are used for the previous Delaunay refinement algorithms do not apply for Algorithm 3 due to the point relocation step. While proving an improved angle bound for it is left open, we can match the earlier bounds by a simple modification to Algorithm 3: Apply the relocation only when the constraint angle is larger than 30°. This modification keeps Algorithm 3 still effective for large constraint angles and provably good for small constraint angles.

3.5 Experiments

We implemented the proposed algorithms and ran experiments on various data sets and point distributions. We include only a handful of output illustrations in this study. (See Figures 3-1, 3-9, 3-10, 3-11, 3-15, 3-16, and 3-17.) However, the performance plots in Figures 3-12, 3-13, 3-14, 3-18, and 3-19 and the statistics in Tables 3-1 and

3-2 reveal much more information regarding the performance of our algorithms in comparison to best known previous methods. Furthermore, Table 3-3 shows the behavior of Algorithm 3 in terms of the types of Steiner points used.

3.5.1 Implementation

Our implementation is a fairly modest modification of Triangle software. Here, we describe the crucial changes we have made. For computing the Type II Steiner points, we have implemented a primitive that computes the intersection(s) of a ray and a circle. This computation is slightly more expensive than computing circumcenters and is common to graphics software [83].

In general, for a Delaunay refinement algorithm, inserting a Steiner point x of a bad triangle pqr (with the shortest edge pq), does not guarantee that pqx is a good triangle. For instance, in the circumcenter insertion algorithm of Ruppert [99], this could happen because $\angle pxq$ can be strictly smaller than α. Whereas, in our algorithm for Type II and Type III Steiner points $\angle pqx$ or $\angle qpx$ can be smaller than α. In our implementation, whenever $\angle pqx$ or $\angle qpx$ is less than α, we opt for using Type IV Steiner point to make sure that pqx is a good triangle.

In the relocation step of Algorithm 3, we avoid computing an exact intersection of the set of petals on the link. Instead, we used an approximate representation of the feasible region of each petal, where the two constraints imposed by the angles incident to two link vertices are also considered. Then, compute an approximate feasible region simply by a half-plane intersection algorithm. Our experiments indicate that using an 8 vertex convex polygon for approximating each such region is efficient and effective. We start the calculations from those that are furthest from each other on the link. If there exists any petal pair with no intersection, then we terminate the process and conclude that no relocation point is found. Otherwise, we compute the centroid of the intersection region b such that all triangles of $star(b)$ in the Delaunay triangulation of the set $(P \cup \{b\} - \{a\})$ are of good quality, where P is the evolving point set and a is the

point to be relocated (see Figure 3-8). We always maintain the Delaunay property of the triangulation. Alternative strategies for point relocation can be explored. We choose this particular one for its simplicity and efficiency.

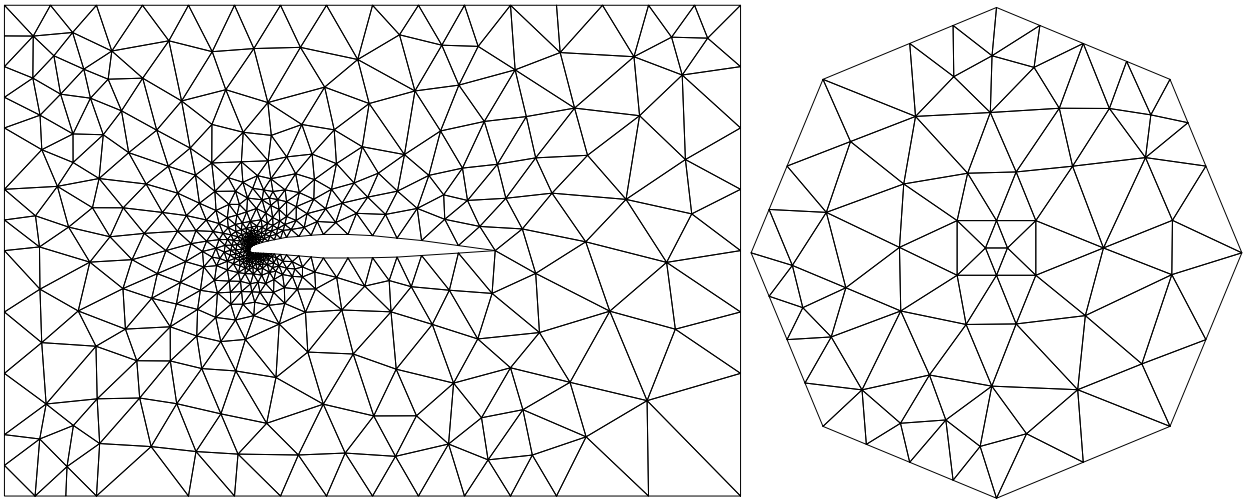


Figure 3-9. Output of the Algorithm 3 for Boeing and octagon data sets for large α values ($\alpha = 42°$). The practical limit ζ of our algorithm for these data sets are actually higher than $42°$ as in Table 3-1.

3.5.2 Data Sets

We have conducted an extensive experimental study on many (over 40) data sets with various properties. Most of the data sets have quite complex boundaries with small angles (e.g., crab, Italy, and Florida) and/or holes in the domain (e.g., butterfly, Superior, and Ishak Pasha castle). We have also tested our algorithms on interesting point set distributions such as random and Kuzmin. Further information (the number of points, the number of segments, and the number of holes) on these data sets can be found in the first four columns of Table 3-1.

3.5.3 Performance Comparison

We compare our algorithms with the previous Delaunay refinement implementations on three main performance measures supported by experimental results.

Table 3-1. Performance of Algorithm 3 with respect to Triangle 1.4 (Ruppert's algorithm) is shown. As additional information, average ζ values for Triangle 1.6 and Algorithm 2 are 35.08 and 38.40, respectively. (Continued on the next page.)

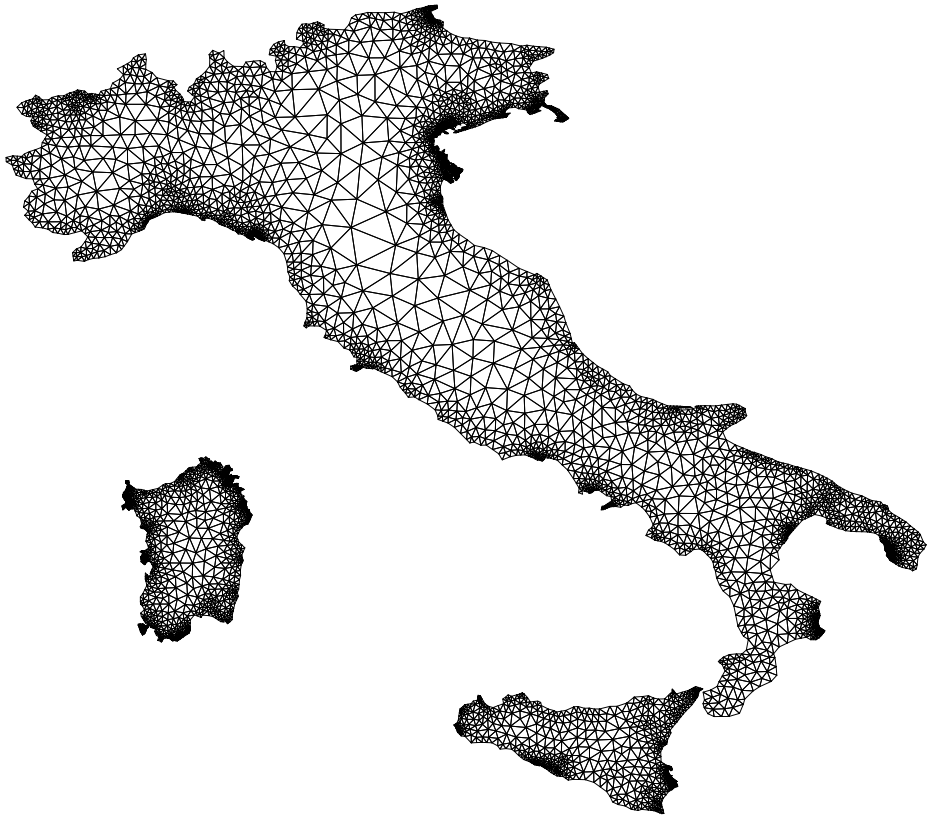
Data set	#Points	#Segments	#Holes	Cut-off angle (ζ) Triangle 1.4	Cut-off angle (ζ) Algorithm 3	Size improvement ($\alpha = 30°$)
Cyprus	448	447	0	33.9	42.3	2.82
Malta	162	162	0	34.4	42.4	2.57
Tri. & Tobago	429	429	0	34.0	42.4	2.78
Turkey	216	216	0	34.1	42.4	2.51
Puerto Rico	77	77	0	34.8	42.4	2.56
Hawaii	1334	1334	0	34.0	42.2	2.96
Italy	1079	1079	0	33.9	42.3	2.72
Japan	7701	7701	0	33.7	42.2	2.78
Florida	304	304	0	33.9	42.4	2.81
California	291	291	0	33.9	42.4	2.68
Texas	641	641	0	33.8	42.3	2.83
New York	282	282	0	34.0	42.4	2.72
Superior	522	522	7	34.2	42.4	1.99
Fish	92	92	1	35.0	42.5	2.10
Kangaroo	112	112	1	35.5	42.4	1.37
Koala	163	163	4	34.5	42.4	1.81
Rabbit	186	186	3	34.4	42.4	1.99
Cheetah	304	304	4	34.4	42.4	1.84
Butterfly	379	379	14	34.2	42.3	2.36
Bat	194	194	3	34.8	42.4	1.88
Dolphin	200	200	1	34.3	42.4	2.37
Goose	234	234	2	34.5	42.4	2.39
Rhea	230	230	1	34.3	42.4	2.00
Goldfish	2685	2685	1	33.8	42.3	2.58
Crab	3168	3168	2	33.8	42.3	2.20
Giraffe	1453	1453	2	34.2	42.4	2.43
Alligator	1100	1100	3	34.0	42.3	2.94
Black cat	236	236	1	34.3	42.4	2.26
Cat	848	848	1	34.2	42.4	2.46
Hexagon	8	6	0	34.8	42.7	1.82
Octagon	10	8	0	36.3	42.6	3.20
Boeing	30	30	1	34.0	42.4	4.14
Boxed pair	6	4	0	34.2	42.6	3.25
Kuzmin	2000	0	0	33.7	42.3	3.19
Plate	65	64	5	34.5	42.4	1.58
Random	1000	0	0	33.8	42.4	2.66

Table 3-1: Continued.

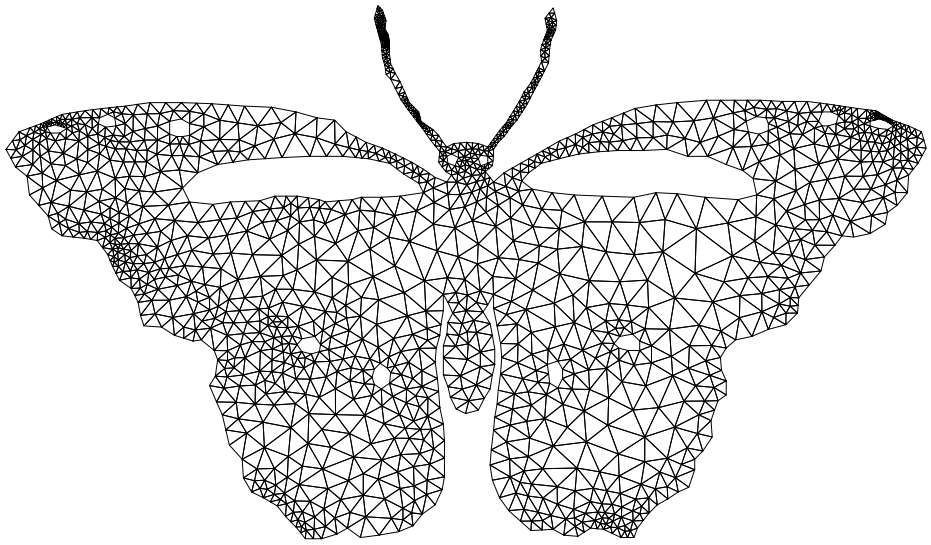
Data set	#Points	#Segments	#Holes	Cut-off angle (ζ) Triangle 1.4	Cut-off angle (ζ) Algorithm 3	Size improvement ($\alpha = 30°$)
Ishak Pasha c.	1650	1650	23	33.8	42.2	2.72
Ayasofia	1374	1374	0	34.0	42.4	2.44
Kız Kulesi	754	754	0	34.1	42.4	2.64
New York view	1241	1241	0	34.1	42.4	2.40
Taj Mahal	379	379	0	34.5	42.4	2.43
Median	**304**	**291**	**N/A**	**34.2**	**42.4**	**2.51**
Average	**844**	**746**	**N/A**	**34.3**	**42.4**	**2.49**

Table 3-2. Number of different types of Steiner points used by four different Delaunay refinement algorithms: Circumcenter insertion, off-center insertion, Algorithm 2 and Algorithm 3. Data set legend: K, Kuzmin; C, California; J, Japan; I, Ishak Pasha castle; D, Dolphin.

Data		CCtr	Off-center		Algorithm 2				Algorithm 3				
set	α	TypeIV	TypeI	TypeIV	TypeI	TypeII	TypeIII	TypeIV	TypeI	TypeII	TypeIII	TypeIV	Rel.
K	20	1895	384	890	328	323	442	3	327	309	348	3	115
K	30	7938	1310	2690	1009	881	1004	41	1012	740	686	33	427
K	34	∞	2443	5535	1470	1445	1556	125	1456	1236	979	80	639
K	37	∞	∞	∞	2117	2421	2498	313	1949	1858	1248	190	944
K	41	∞	∞	∞	∞	∞	∞	∞	3648	4091	2615	719	2057
C	20	261	75	78	63	45	39	0	63	34	28	0	22
C	30	792	171	286	119	120	89	1	120	111	53	3	37
C	34	∞	298	532	163	216	120	7	170	180	79	3	51
C	37	∞	∞	∞	255	379	230	14	216	299	108	4	95
C	41	∞	∞	∞	∞	∞	∞	∞	393	573	198	39	160
J	20	5049	1196	2170	1033	986	952	7	1065	859	701	9	378
J	30	18493	3532	6148	2479	2871	1953	60	2550	2404	1338	34	971
J	34	∞	6462	11793	3500	4626	2972	157	3473	3905	1887	97	1345
J	37	∞	∞	∞	4956	7302	4758	468	4680	5765	2623	221	1881
J	41	∞	∞	∞	∞	∞	∞	∞	9859	14178	7143	1698	5834
I	20	1596	447	747	378	300	411	2	402	249	293	4	178
I	30	4961	1111	1815	760	785	509	14	755	697	359	2	240
I	34	∞	1835	3327	963	1299	788	42	997	1086	525	19	355
I	37	∞	∞	∞	1331	1960	1317	125	1299	1608	755	52	479
I	41	∞	∞	∞	∞	∞	∞	∞	2392	3306	1519	390	1168
D	20	120	32	54	31	12	33	0	31	11	26	0	12
D	30	329	76	136	55	47	53	3	57	39	41	2	27
D	34	1312	105	249	67	74	82	4	68	65	56	1	34
D	37	∞	∞	∞	88	110	98	11	89	94	56	6	37
D	41	∞	∞	∞	∞	∞	∞	∞	144	212	124	28	103



A



B

Figure 3-10. Output of the Algorithm 3 for two complex data sets, Italy and butterfly, for large α values ($\alpha = 41^\circ$ and $\alpha = 41.5^\circ$). The practical limit ζ of our algorithm for these data sets are actually higher than 41° and 41.5°, respectively as in Table 3-1.

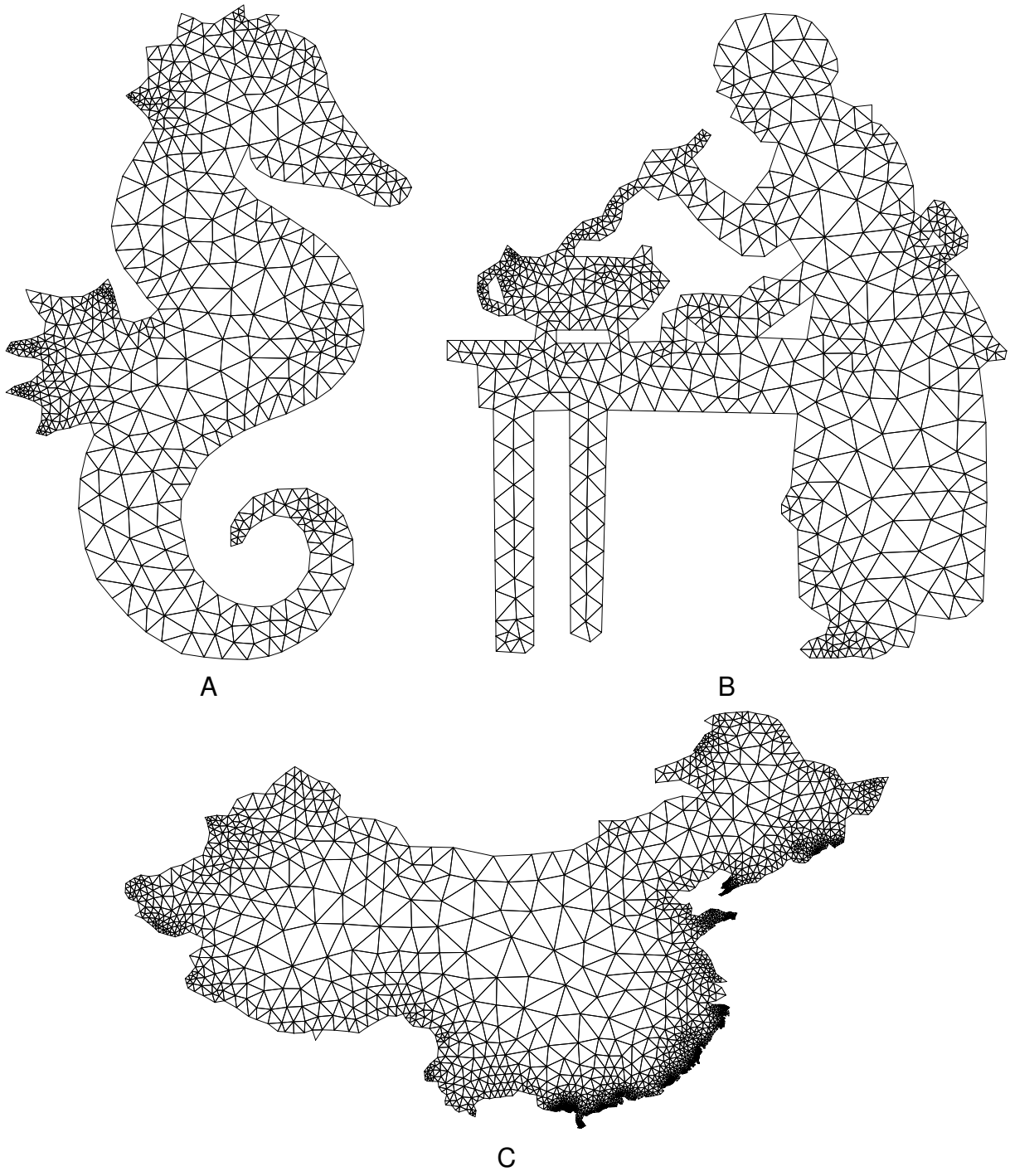


Figure 3-11. Output of the Algorithm 3 for three complex data sets, cooking lady, seahorse and China, for large α values ($\alpha = 41^{\circ}$). The practical limit ζ of our algorithm for these data sets are actually higher than 41°.

Table 3-3. Percentage values of different types of Steiner points used by Algorithm 3. Data set legend: K, Kuzmin; C, California; J, Japan; I, Ishak Pasha castle; D, Dolphin.

		Algorithm 3					
Data set	α	TypeI	TypeII (A)	TypeII (B)	TypeII (Rest)	TypeIII	TypeIV
K	20	33.1	26.2	4.0	1.1	35.3	0.3
K	30	41.0	22.4	5.8	1.8	27.8	1.3
K	34	38.8	25.4	6.5	1.1	26.1	2.1
K	37	37.2	27.9	6.8	0.8	23.8	3.6
K	41	32.9	30.6	6.2	0.1	23.6	6.5
C	20	50.4	25.6	0.8	0.8	22.4	0.0
C	30	41.8	33.4	4.2	1.0	18.5	1.0
C	34	39.4	36.3	4.2	1.2	18.3	0.7
C	37	34.4	41.9	5.6	0.2	17.2	0.6
C	41	32.7	43.6	4.0	0.0	16.5	3.2
J	20	40.4	30.9	1.0	0.8	26.6	0.3
J	30	40.3	33.6	2.8	1.6	21.2	0.5
J	34	37.1	36.2	4.1	1.4	20.2	1.0
J	37	35.2	37.4	5.1	0.9	19.7	1.7
J	41	30.0	37.0	6.0	0.1	21.7	5.2
I	20	42.4	25.1	0.9	0.2	30.9	0.4
I	30	41.6	36.9	1.4	0.1	19.8	0.1
I	34	38.0	39.0	2.1	0.3	20.0	0.7
I	37	35.0	39.2	3.7	0.4	20.3	1.4
I	41	31.4	38.5	4.9	0.1	20.0	5.1
D	20	45.6	13.2	1.5	1.5	38.2	0.0
D	30	41.0	25.9	1.4	0.7	29.5	1.4
D	34	35.8	30.5	3.2	0.5	29.5	0.5
D	37	36.3	33.1	4.5	0.8	22.9	2.4
D	41	28.3	36.8	4.9	0.0	24.4	5.5

Large constraint angles. Plots in Figures 3-12, 3-13, and 3-14 as well as the fifth column of Table 3-1 show that the original Delaunay refinement algorithm is impotent for constraint angles α larger than the cut-off angle ζ for this method which is $34°$. The off-center insertion algorithm of Üngör has already extended ζ to $35°$. On the other hand, our Algorithm 2 terminates with correct output for constraint angles up to $38.5°$. Finally, our Algorithm 3 works for constraint angles up to $42°$.

Note that plots in Figures 3-12, 3-13, and 3-14 are composed of two different parts, mainly demonstrating ζ. The upper part plots the number of Steiner points used by

each of the four algorithms. The lower part plots the ratio of the Steiner points used by Triangle 1.4 and Algorithm 3. All plots in both parts are with respect to the same x-axis, that is, the constraint angle. Our experiments are done for every $0.1°$, $0.1°$, and $0.5°$ angles for the given range in Figures 3-12, 3-13, and 3-14, respectively. Plots in the upper parts clearly show the cut-off angles, ζ values, for each of the four algorithms.

Tables 3-2 and 3-3 list the distribution of each type of Steiner point used by the algorithms and might help explain the improvements achieved by our algorithms. Statistics reveal that, among the four type of Steiner points, circumcenters are least frequently used by our algorithms. There are cases where circumcenters are altogether avoided. In general, each of the other types constitute around 30 to 35% of the total Steiner points, only at most 7% of them are circumcenters. As we increase the minimum angle constraint, we observe that circumcenters are chosen more frequently. In addition, Lemma 2 and Table 3-3 together further clarify our results. It is also interesting to note that the number of point relocations performed is quite small compared to the total number of Steiner points. It seems that only around 5% of the Steiner points are relocated when the constraint angle is small ($20°$ or less). This percentage reaches around 15% for larger constraint angles. See the last column of Table 3-2.

Output size. The number of triangles in a triangulation is a simple linear function of the number of points in it. Hence, the plots in Figures 3-12, 3-13, and 3-14 reflect on the number of triangles in the output. We observed that the two algorithms proposed here give significant output size improvements over the previous algorithms. These improvements are particularly impressive when the constraint angle is large. In Figures 3-12 and 3-13, we highlight the size improvements for constraint angles $30°$ and larger, whereas Figure 3-13 presents the plots for the entire range of constraint angles. Size ratio plots, the bottom parts of the figures, show the behavior of this improvement while the constraint angle changes. (See Figure 3-15, 3-16, and 3-17 also.)

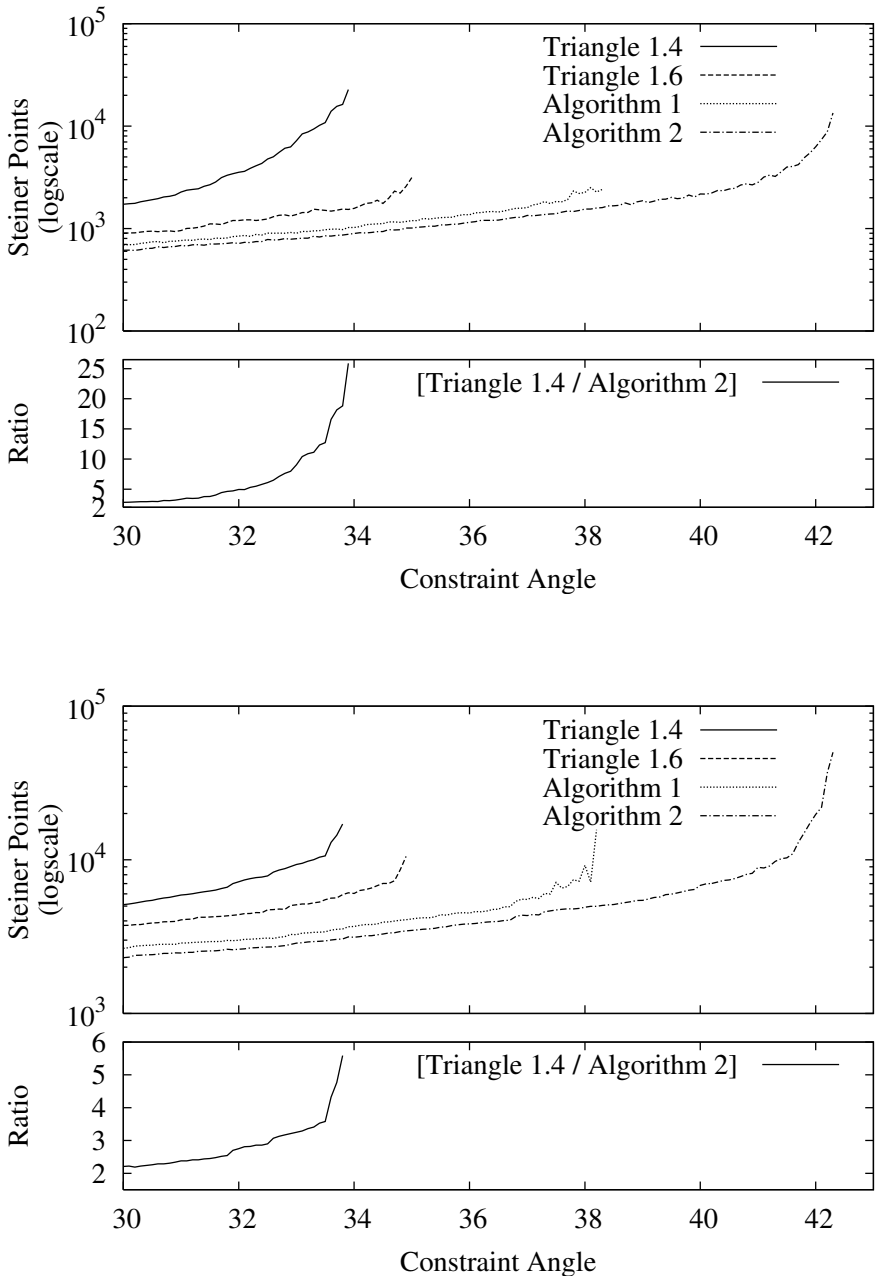


Figure 3-12. Output size comparison plots for two data sets (Cyprus and crab), which also reveal the practical angle bounds of the four algorithms, demonstrating ζ.

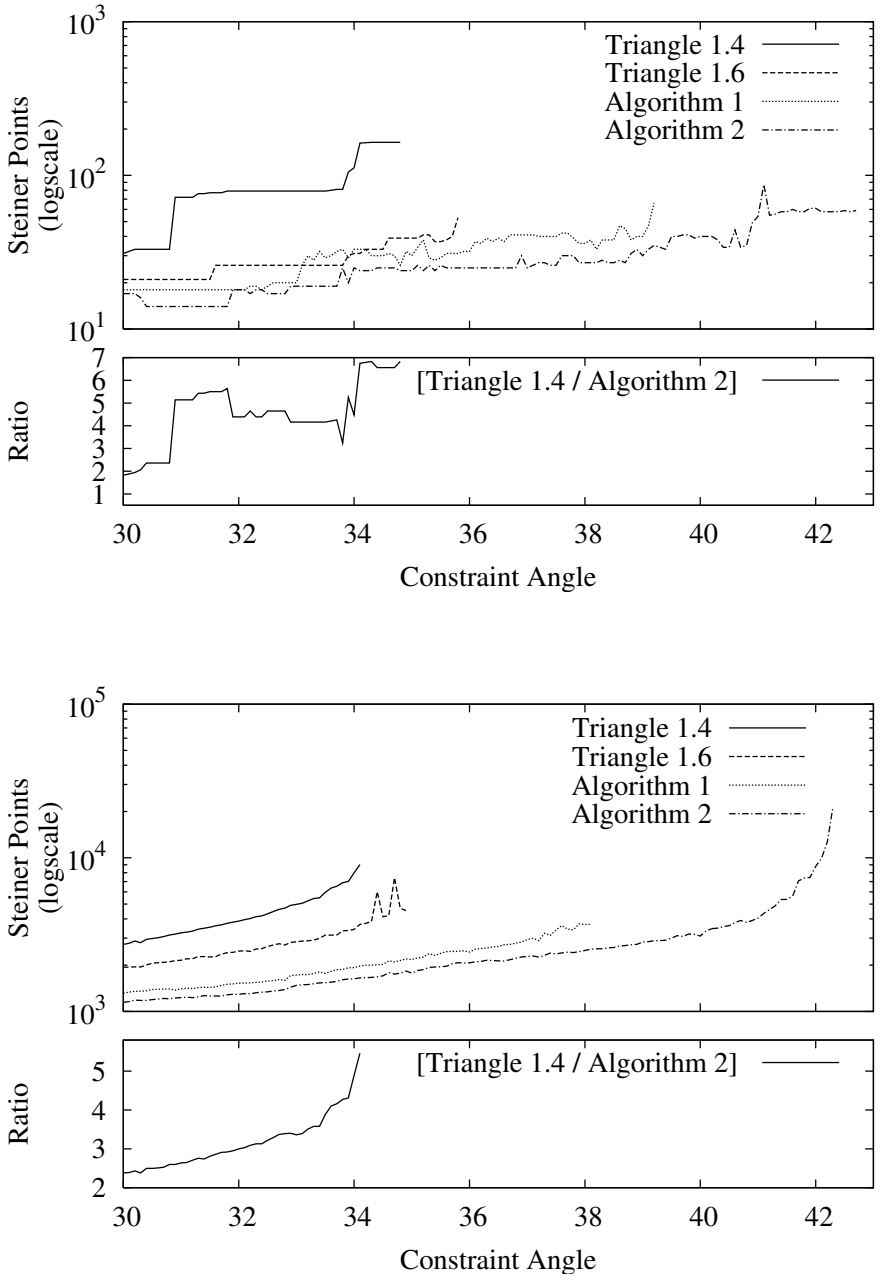


Figure 3-13. Output size comparison plots for two data sets (hexagon and New York view), which also reveal the practical angle bounds of the four algorithms, demonstrating ζ.

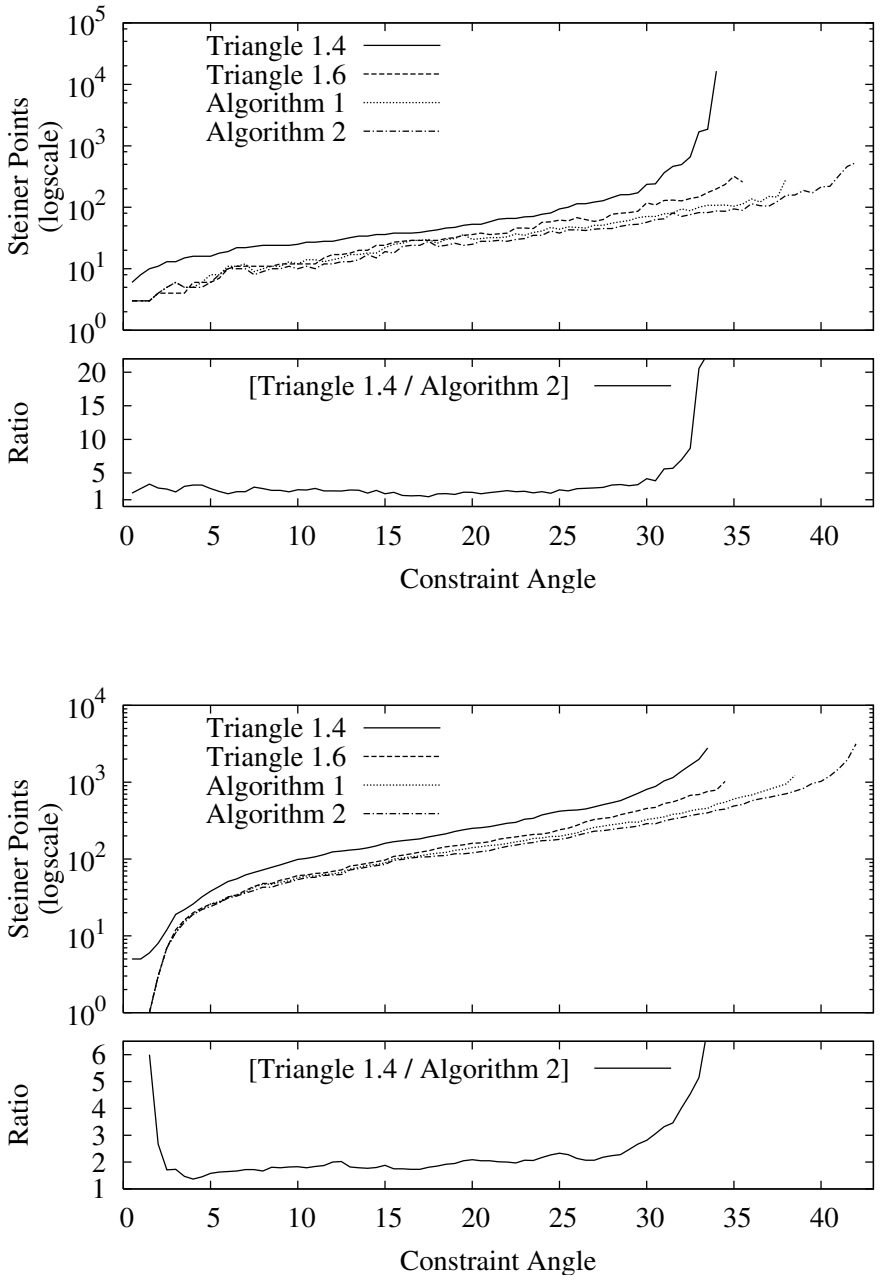


Figure 3-14. Output size comparison plots for two data sets (Boeing, Florida), which also reveal the practical angle bounds of the four algorithms, demonstrating ζ. Triangle 1.4 inserts additional points on the boundary, even if no point insertion is necessary for satisfying the constraint angle α. Large size improvement observed for small α values is a simple outcome of this difference.

Furthermore, the last column of Table 3-1 indicates the ratio of the number of Steiner points inserted by Triangle 1.4 over Algorithm 3 for $\alpha = 30°$. On average, the size of the triangulations decreased around a factor of 2. In addition, Figure 3-18 represents a more extensive version of this experiment which has been completed by using around 40 data sets, constructing quality triangulations of those, for every $1°$ angle in between $5°$ and $35°$ by using Triangle 1.4, Triangle 1.6, and Algorithm 3. Based on this plot, the output size improvements of our algorithm in comparison to both previous algorithms can be classified as "significant," "good," and "impressive" in the constraint angle ranges [5°–20°), [20°–30°), and [30°–35°], respectively. This plot reveals, for instance, that our algorithm gives around 50% smaller meshes (average over 40 data sets) than the best known previous Delaunay refinement implementation when the constraint angle is $30°$.

Running time. The primitives we use for computing the proposed locally optimal Steiner points are slightly more involved computationally than those used for computing circumcenters. However, we insert fewer Steiner points than the previous algorithms. Overall, both Algorithms 2 and 3 run faster than the previous algorithms. Between the two of our algorithms, Algorithm 3 is slightly faster than Algorithm 2. (See Figure 3-19.)

3.5.4 Results on Uniform Triangulations

There are many scientific applications where the additional constraints are imposed for the quality Steiner triangulation problem. We mentioned earlier that the (shape) quality of a mesh element can be a crucial factor on the interpolation errors introduced during the use of these meshes in simulations. Another factor it turns out is the size of the mesh elements. It can be noted in the experiments shown previously in Section 3.5 that the size of the triangles in a graded mesh is dictated by the domain geometry description together with the constraint angle. For most applications, further constraints need to be imposed in order to contain the errors in the simulation at a reasonable level.

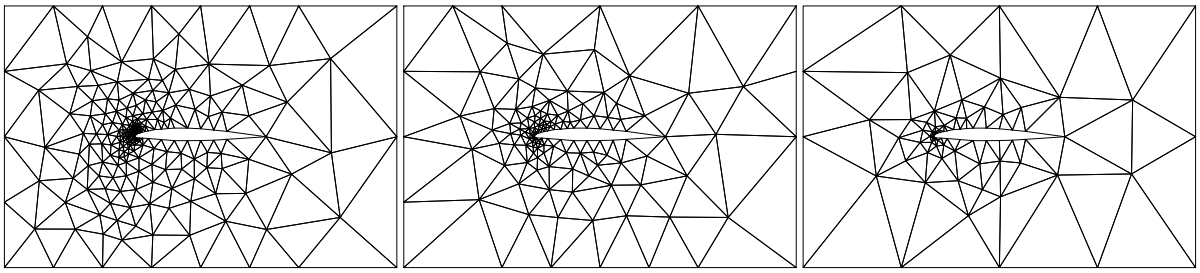


Figure 3-15. Output size comparison on the Boeing data set. A) Circumcenter method (Triangle 1.4). B) off-center method (Triangle 1.6). C) Algorithm 3. For $\alpha = 30°$, the new algorithm inserts 57 Steiner points, almost half as many as the 118 Steiner points inserted by the off-center algorithm which is in turn half as many as the 236 Steiner points used by the circumcenter insertion algorithm. The output of the Algorithm 2 looks very similar to that of Algorithm 3, with 71 Steiner points.

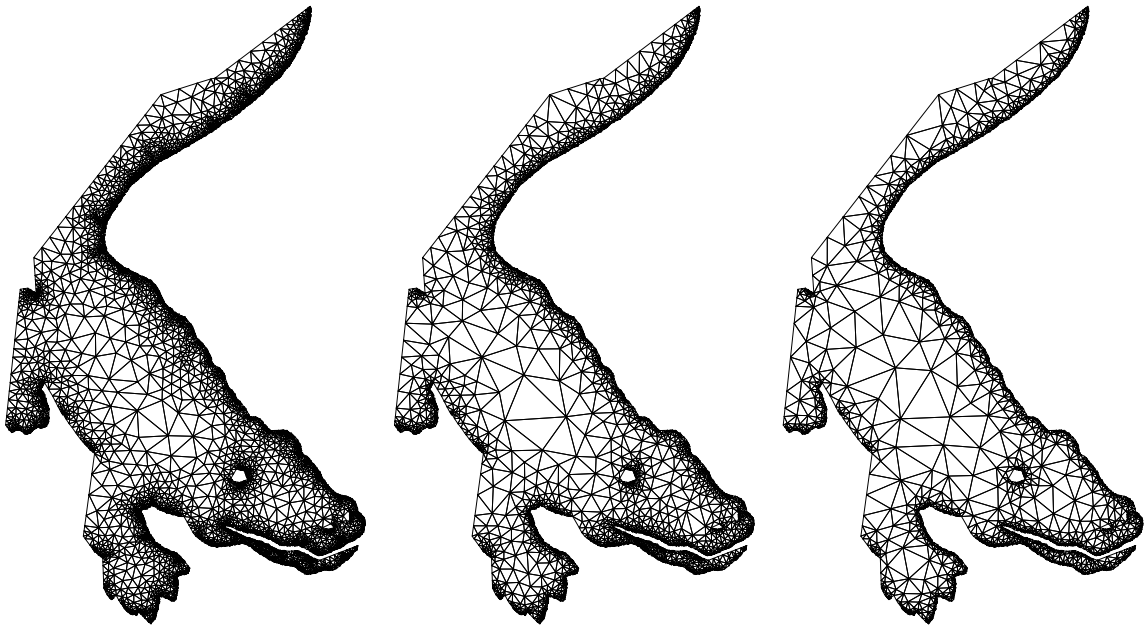


Figure 3-16. Output size comparison on the Alligator data set. A) Circumcenter method (Triangle 1.4). B) off-center method (Triangle 1.6). C) Algorithm 3. For $\alpha = 30°$, the new algorithm inserts much fewer Steiner points (1376) than the off-center and the circumcenter insertion algorithms (2163 and 4976 respectively). The output of the Algorithm 2 looks very similar to that of Algorithm 3, with 1564 Steiner points.

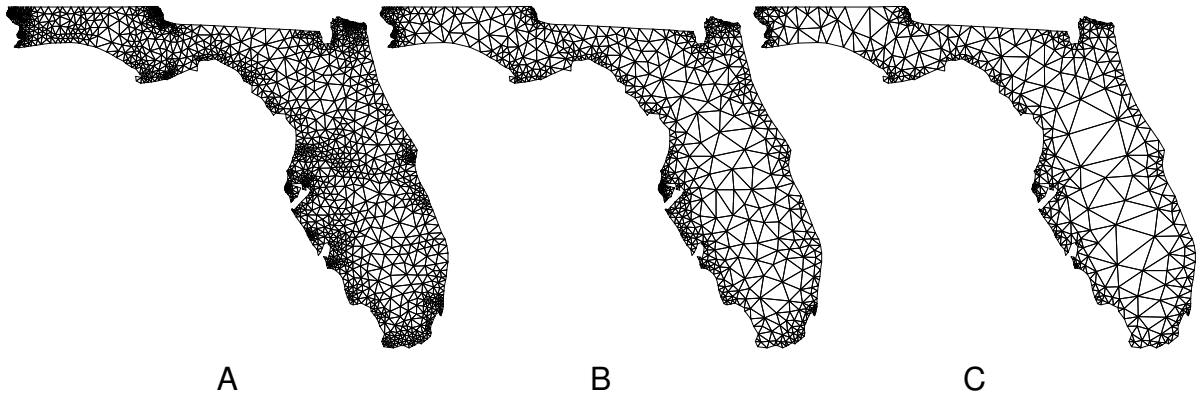


Figure 3-17. Output size comparison on the Florida data set. A) Circumcenter method (Triangle 1.4). B) off-center method (Triangle 1.6). C) Algorithm 3. For $\alpha = 32°$, the new algorithm inserts much fewer Steiner points (346) than the off-center and the circumcenter insertion algorithms (606 and 1391 respectively). The output of the Algorithm 2 looks very similar to that of Algorithm 3, with 392 Steiner points.

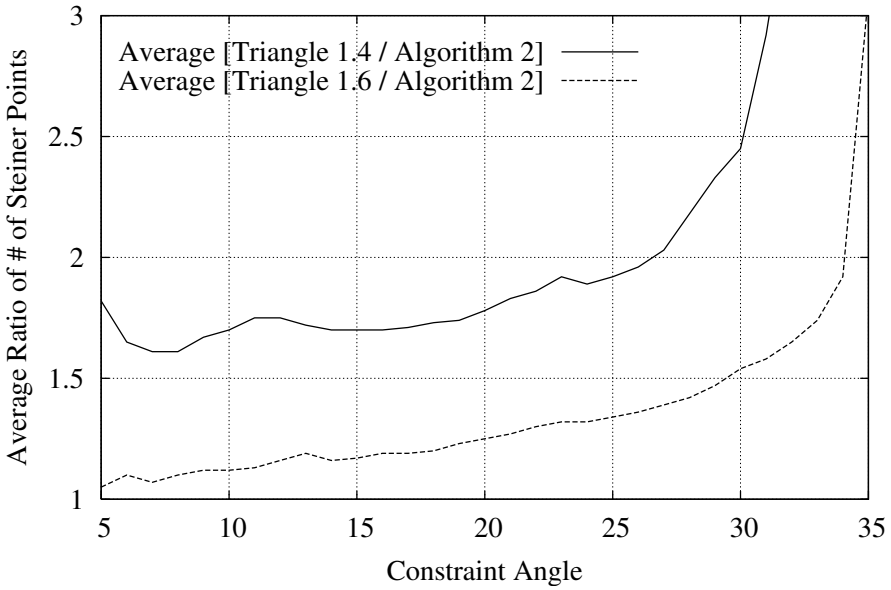


Figure 3-18. Plot of average output size ratio of Triangle 1.4 and Triangle 1.6 with Algorithm 3 in terms of the number of Steiner points vs. the constraint angle α for all data sets shown in Table 3-1.

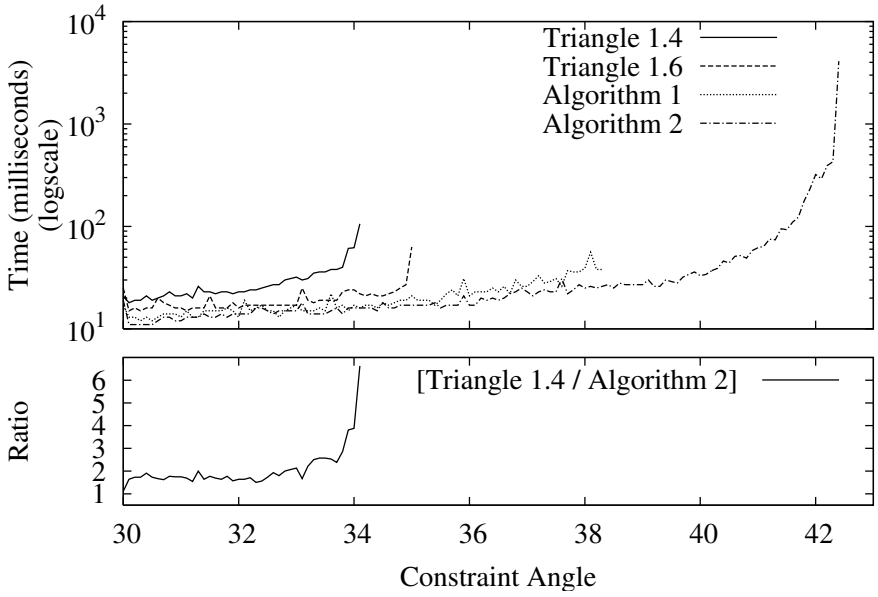


Figure 3-19. The running times of four different algorithms are plotted (with respect to varying constraint angle) for Turkey data set.

A commonly used mesh element size constraint is the maximum area constraint which often results in uniform meshes. Triangle software, for instance, allows the user to specify the maximum area constraint. Unfortunately, the additional constraint does not help Triangle to overcome the termination problem. As is the case in graded meshing, the software gets into an infinite loop when the constraint angle is larger than $34°$. Below, we summarize our results on the performance of the proposed algorithms when they are used for computing uniform meshes.

Angle bounds. The additional area constraint does not deteriorate the practical angle bounds of our algorithms. As illustrated in Figures 3-20, 3-21, 3-22 and 3-23, Algorithms 2 and 3 generally terminate for constraint angles around $38.5°$ and $42°$, respectively, even when a maximum area constraint is imposed. These ζ values are consistent with our previous results over data sets given in Table 3-1.

Output size. A surprising outcome of our experiments in uniform meshing was the improvements on the size of the output. After all, the maximum area constraint clearly

dictates the size of each element in the mesh. So, we think it is remarkable that we observe significant size improvements (in the range of 10 to 50%). Figure 3-24 illustrates that Algorithm 3 creates a much sparser triangulation than the previous algorithms under the same minimum angle and maximum area constraints. It seems that we achieved output size improvement (for uniform triangulation) by creating triangles whose areas are usually quite close to the specified area constraint.

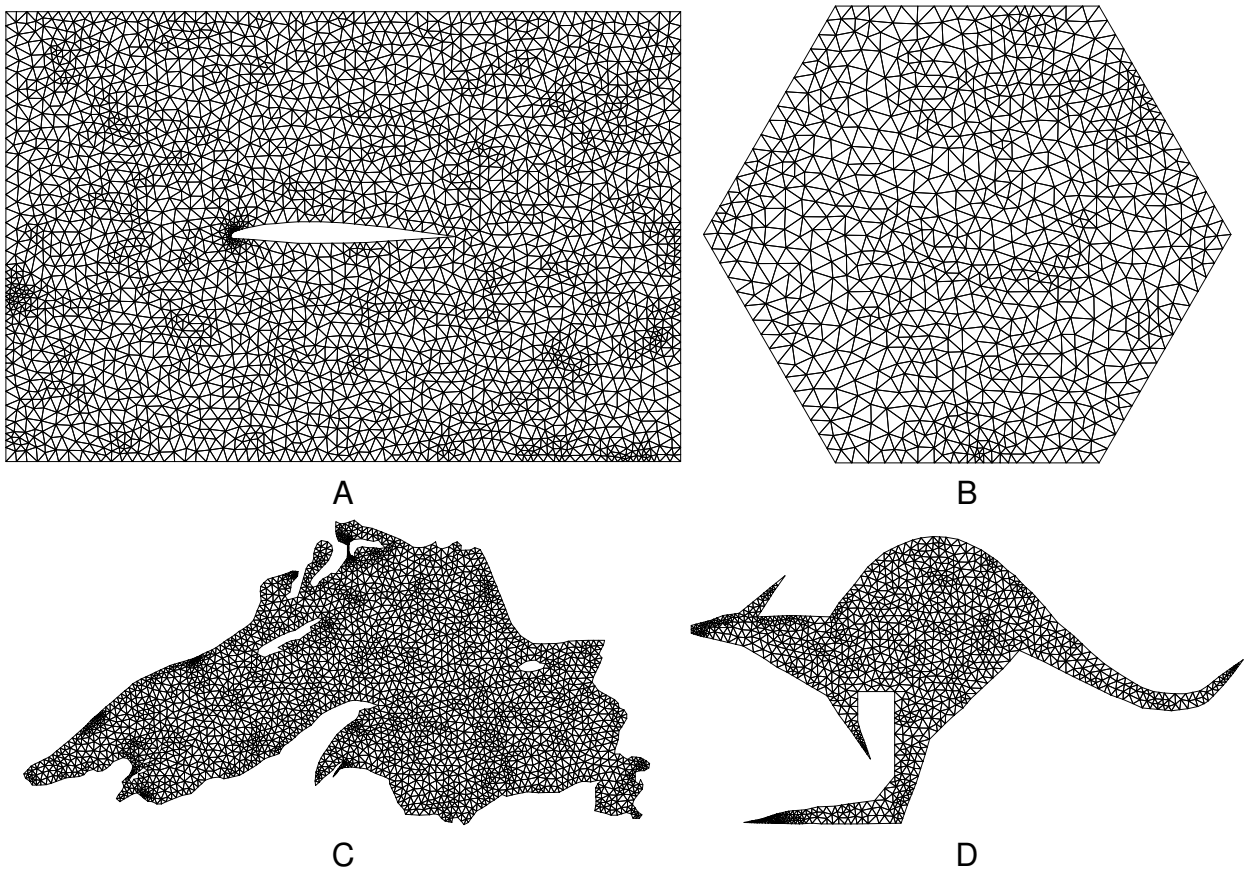


Figure 3-20. Output of Algorithm 3 for various data sets for various large α values for a fixed maximum area value. A) $\alpha = 41°$, max area $= 0.002$, shortest edge $= 0.0036$. B) $\alpha = 41.5°$, max area $= 0.05$, shortest edge $= 1$. C) $\alpha = 41°$, max area $= 0.00006$, shortest edge $= 0.0018$. D) $\alpha = 41°$, max area $= 20$, shortest edge $= 3.12$. These α and maximum area values should not be seen as the limit of our algorithm for the shown data sets. Values have been chosen accordingly to have clear example figures. Also, the shortest edge length in Delaunay triangulation of these data sets are given as references for maximum area values. For a better idea, see the plots in Figures 3-23 and 3-22.

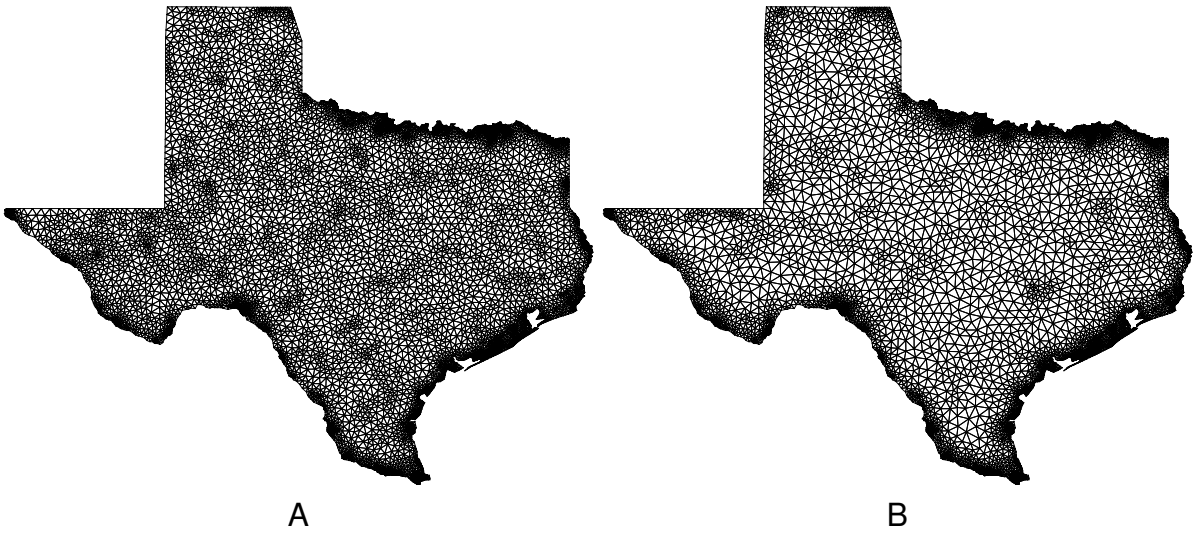


Figure 3-21. Output of Algorithm 3 for Texas data set for different maximum area constraints. A) $\alpha = 41.5°$, max area $= 150$, shortest edge $= 1$. B) $\alpha = 41.5°$, max area $= 325$, shortest edge $= 1$.

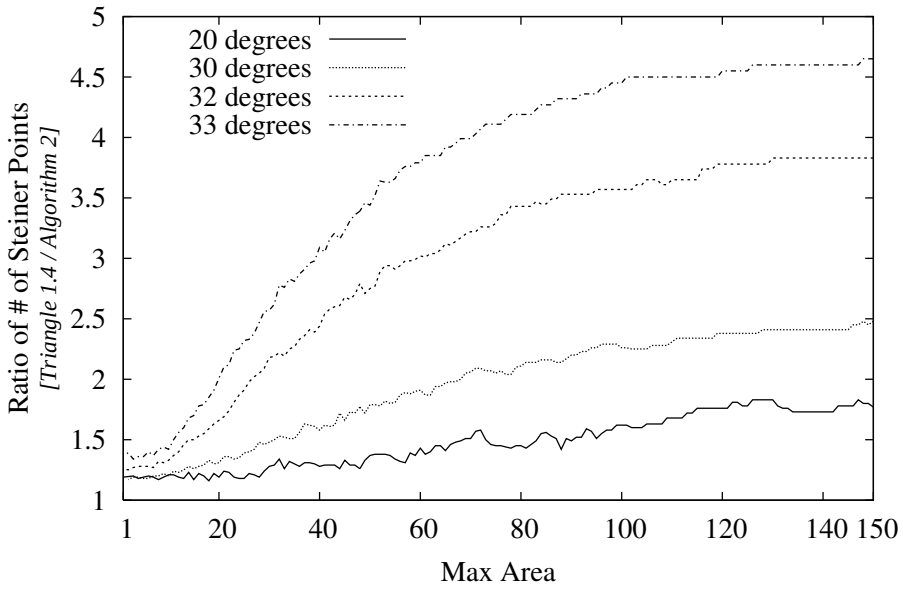


Figure 3-22. Plot of the output performance ratio for the Puerto Rico data set having unit distance shortest edge length. The ratio represents the number of Steiner points inserted by Triangle 1.4 over the corresponding number of points inserted by Algorithm 3 for different constraint angles for some varying maximum area constraint. Note that the ratio is always greater than 1.15 showing that Algorithm 3 outperforms Triangle 1.4 even with a small area constraint.

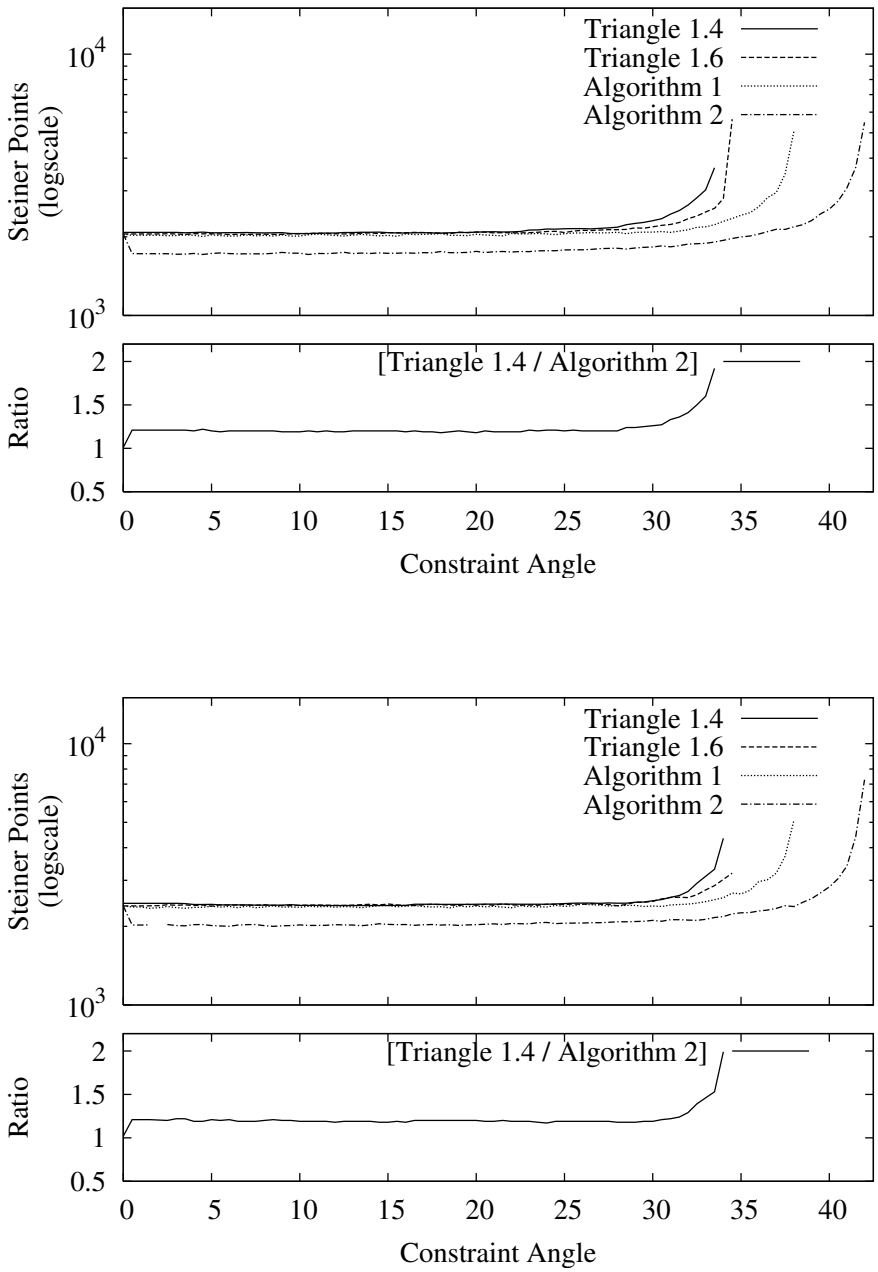
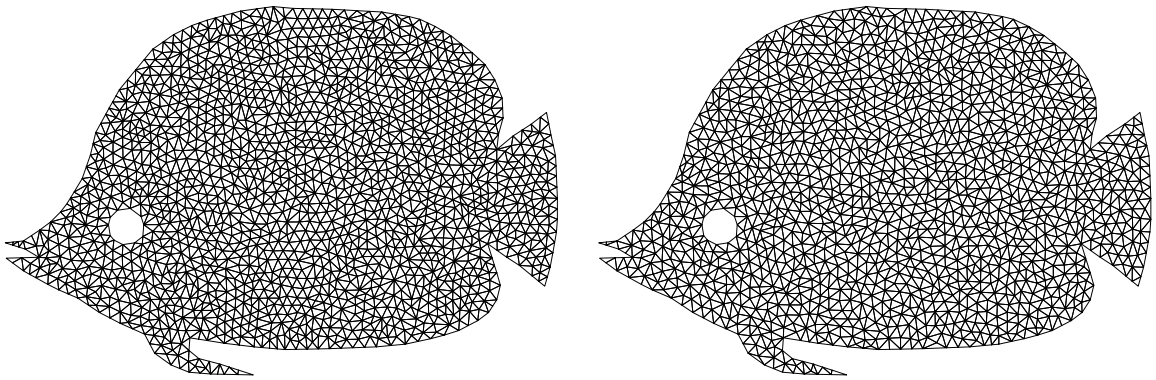


Figure 3-23. Plot of the number of Steiner points vs. the constraint angle α for some fixed maximum area constraint. The maximum area constraint value ($= 50$) for the Florida data set is chosen as 50 times the square of the shortest edge ($= 1.0$), whereas for the Turkey data set the maximum area constraint ($= 130$) is 10 times the square of the shortest edge ($= 3.6$) of Delaunay triangulations of each data sets.



A B

Figure 3-24. Output size comparison on the fish data set. A) Triangle 1.4 (circumcenter). B) Algorithm 3. For $\alpha = 20^\circ$ and maximum area $= 18$, which is double of the square of shortest edge in Delaunay triangulation of the data set (shortest edge $= 2.98$), the new algorithm inserts 1310 Steiner points, around 17 percent less than the 1572 Steiner points inserted by the circumcenter algorithm. The off-center algorithm inserts 1584 Steiner points, which is almost the same as the circumcenter algorithm. The average triangle areas for the output of Triangle 1.4 and Algorithm 3 are 11.44 and 13.87, respectively.

Performance. We should note that the relative performance of the algorithms depends on which constraint dominates more. The behaviors of the constraints can be easily observed from Figure 3-22 where the plot shows the comparative behavior of Algorithm 3 and Triangle 1.4 based on changes in the maximum area constraint in terms of output size ratio as well as the effect of having different constraint angle α values.

A number of evident observations from the figure can be summarized as follows: When the maximum area constraint is between 1 and 10, changing α has little effect but changing maximum area in this range, for fixed α, also has little effect (where the maximum area constraint is dominant). For maximum area above 20, changing α for a fixed maximum area has a significant effect (where the constraint angle is dominant). For maximum area above 100, changing maximum area for a fixed α has little effect (where the constraint angle is still dominant). Overall, as we choose a larger area constraint, size improvement over Triangle 1.4 increases moderately and becomes constant. In addition, increasing the constraint angle α affects the amount of size

improvement significantly. Notice that our algorithm still performs notably better, even if the maximum area constraint is dominant, forcing the algorithms to create uniform elements.

Similarly, Figure 3-23 shows the overall behavior of different versions of Triangle together with Algorithms 2 and 3 for a fixed value of the maximum area constraint over the complete range of angles in terms of the number of Steiner points at the upper parts of the plots, while the size ratio between Triangle 1.4 and Algorithm 3 can be seen at the bottom. Clearly, compared to previous versions of the Delaunay refinement algorithms, our algorithms produce smaller-sized triangulations. Also note that the ratio stays significantly above 1, which indeed corresponds to a size improvement around 15% on average. Naturally, the improvements are more significant, for larger α values, as the constraint angle increases.

3.5.5 Comparison: Smoothing Methods

We also compared the performance of post-refinement-smoothing methods with our method. In Figure 3.5.5, we applied smart Laplacian and angle-based smoothing on an initial quality triangulation ($\alpha = 30^\circ$) for Africa data set generated by circumcenter insertion strategy. As can be seen, the angle quality is slightly improved or even decreased after smoothing operation. However, our method generates higher quality triangulation ($\alpha = 40^\circ$) with nearly the same number of points in half of the time it takes for classical Delaunay refinement. In addition, applying smoothing to this triangulation only helps to smooth the angle values and deteriorate or does not improve the angle quality. This indicates that our integrated relocation and refinement strategy is more powerful and significantly faster than the combination of generating a good quality mesh and applying smoothing afterwards.

It is important not to confuse our algorithm with a global mesh optimization technique [3, 35, 60]. Such mesh optimization algorithms can give further improvements on the output of both our algorithms and the previous Delaunay refinement algorithms.

Here, our brief experiments on the output of popular smoothing algorithms indicate that we get smoother angle distribution/histograms, that is, a post-refinement-smoothing help decreasing the number of angles close to the constraint angle. However, a significant improvement is rarely observed on the extreme angles (the smallest and the largest). There are many cases where the smallest angle deteriorates with smoothing.

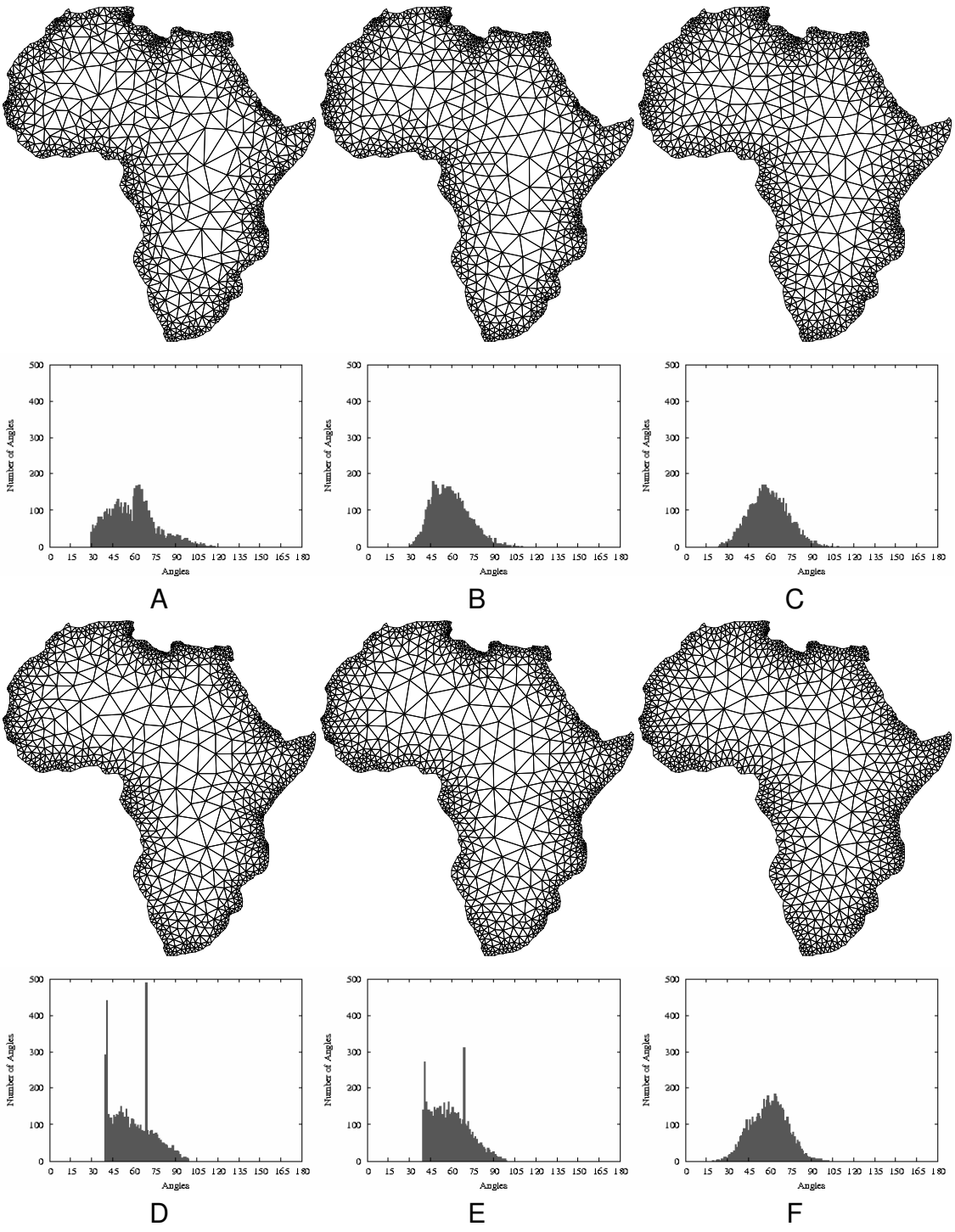


Figure 3-25. Comparing smoothing algorithms with our method. A) Triangle 1.4 (circumcenter), $[\alpha, \gamma] = [30°, 118°]$, 1073 points, 83 msec. B) Smart Laplacian smoothing applied on (A), $[\alpha, \gamma] = [31.2°, 110.8°]$, 348 msec. C) Angle-based smoothing applied on (A), $[\alpha, \gamma] = [24.2°, 109.8°]$, 346 msec. D) Algorithm 3, $[\alpha, \gamma] = [40°, 99.8°]$, 1086 points, 40 msec. E) Smart Laplacian smoothing applied on (D), $[\alpha, \gamma] = [40°, 99.3°]$, 347 msec. F) Angle-based smoothing applied on (D), $[\alpha, \gamma] = [20.5°, 102°]$, 377 msec.

CHAPTER 4
TRIANGULATIONS WITHOUT SMALL AND LARGE ANGLES

4.1 Introduction

Having very large (obtuse) angles or very small angles in a triangulation often leads to undesired numerical results in scientific simulations [6, 105]. A lower bound α on the angles of a triangulation implies an immediate upper bound $\pi - 2\alpha$ on its angles. Most triangulation (meshing) algorithms and software rely on this observation and provide a lower bound on the angles.

While for many applications, such an upper bound (immediate from a lower bound) serves as part of a satisfactory solution, there are many cases it leads to sub-optimal numerical solutions or even worse, do not meet the requirements of the numerical formulation in use. There are triangulation algorithms providing directly an upper bound on the angles [9, 12, 13, 15, 49, 86]. However, in general an upper bound do not imply a lower bound on the angles of a triangulation. Except only a few studies addressing the problem [7, 82], there has been a lack of triangulation methods and software that provide strong lower and upper bounds on the angles simultaneously. Here, we propose a method that triangulates complex domains such that all angles are in an interval as narrow as $[40°, 86°]$, addressing Problem 1 in Section 2.3.

4.1.1 Our Idea and Contribution

Table 4-1. A qualitative comparison of our method against the previous methods, where we label the performance of each method on a scale of 4, as 1. Poor, 2. Fair, 3. Good, or 4. Premium.

Feature	Classical Delaunay refinement [99, 104]	LOSP method [51]	WCT method [120]	Acute method [50]	Proposed method
Time Performance	Good	Good	Poor	Fair	Good
Handling complex input	Good	Good	Fair	Good	Good
No small angle	Good	Premium	Fair	Fair	Premium
No large angle	Fair	Fair	Good	Good	Premium
No large no small angle	Fair	Fair	Fair	Fair	Premium

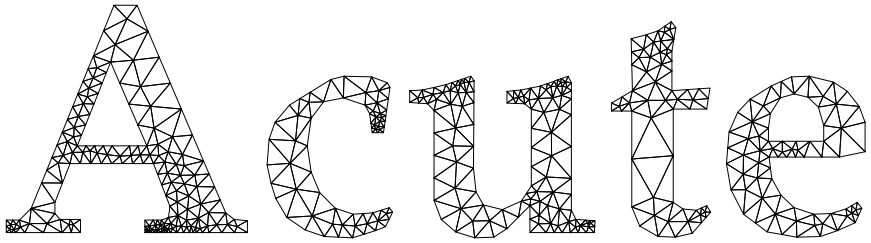


Figure 4-1. Acute triangulation ($[\alpha, \gamma] = [25, 85]$) of the domain **Acute** (in Palatino font) with no small angle $< 25°$.

Table 4-1 gives a practical performance comparison of the proposed method and four available implementations, where each relies on a different approach. Note that, quadtree-based methods are omitted, since they are expected to generate large triangulations in practice. Qualitative labels assigned in this table and related contributions are justified by the experimental study given in Section 4.4. Contribution of Chapter 4 can be summarized as:

- Strong simultaneous lower and upper bounds on angles in practice.
- Fast premium quality triangulation software handling complex geometry.
- Output size similar to Delaunay refinement algorithms.
- Premium quality graded or uniform triangulations.

Our approach for solving the $[\alpha, \gamma]$-triangulation problem can be classified as a Delaunay refinement algorithm. We maintain a Delaunay triangulation while iteratively inserting Steiner points in order to eliminate the triangles with undesired angles. The proposed algorithm, however, is considerably different than the classical Delaunay refinement algorithms of [28, 45, 99, 104, 118] and also the time-efficient Delaunay refinement algorithms of [68, 84]. First, we employ a generic optimization-based Steiner point insertion procedure rather than using a particular type of Steiner point (such as circumcenters, sinks, or off-centers). Second, we integrate a local point relocation procedure into the refinement (in order to avoid the termination problem which is a common undesired never-ending refinement syndrome when strong angle constraints

are imposed). Both of these ideas were shown to be effective already for solving the no small angle triangulation problem in Chapter 3 and in [53]. Here, we consider both the maximum angle and the minimum angle constraints (simultaneously) and describe a new Steiner point insertion strategy and a new point relocation procedure. Next, we introduce geometric concepts that enable us integrating the maximum angle constraint into the refinement and the relocation steps of our modified Delaunay refinement method.

4.1.2 Motivation

Angle constraints are often motivated by numerical methods that make use of triangulations in scientific simulations [6, 13, 15, 105]. Existing methods, however, generally provide either a lower or an upper bound on the angles (and not strong enough lower and upper bounds). Here, our goal is to design a method and a software for triangulating complex geometry providing strong lower and upper bounds on angles simultaneously. This, in turn, leads to a reduction of numerical and interpolation errors that appear in the use of triangulations in scientific computing. Among several versions of the angle bounded triangulation problems, two are worth to elaborate as they ask for relatively strong upper bounds on angles: the non-obtuse triangulation problem and the acute triangulation problem. There are quite a few scientific and graphics applications stated in the literature motivating these two problems. Examples of these applications include the space-time discontinuous Galerkin methods [119] and geodesic path computation on manifolds [71, 96], mesh embedding via discrete Harmonic maps [42], discrete exterior calculus methods [120] and discrete maximum principle on linear finite element solutions [72]. Note that, the lower the upper bound on large angles, the higher the efficiency, accuracy and validity of the applications. In particular, they mostly require acute triangulations. In general, having an additional lower bound on the angles is also implicitly important in these applications.

As we described in Section 2.3, there are many algorithms for computing triangulations with no small angles, e.g., the quadtree-based methods [13], the Delaunay refinement methods [53, 68, 84, 99, 104], and the edge refinement methods [98]. Among these arguably the best performance is due to Delaunay refinement methods, which in practice work for α as high as 34°. They, however, suffer from a termination problem even for simple geometric input domains whenever the desired lower bound α is 35° or larger. Only recently, our new Delaunay refinement algorithm given in Chapter 3 and in [53] that relies on locally optimal Steiner points is shown to work in practice for α as high as 42°. Such an α bound can keep all angles in the interval $[42^\circ, 96^\circ]$, however, it is not strong enough for computing acute or non-obtuse triangulations. So, even the best existing refinement methods do not give a satisfactory solution for Problem 1. Here, we elaborate more on locally optimal Steiner point concept together with the local relocation strategy and provide triangulations without small and large angles.

4.2 α-petal, γ-petal, α-wedge, γ-slab, $[\alpha, \gamma]$-slice

A triangle is said to be good, if its all three angles are in the interval $[\alpha, \gamma]$. Otherwise, the triangle is considered to be bad. Let pqr be a bad triangle in a triangulation and pq be its shortest edge. See Figure 4-2. Consider the circle that goes through p, q and a third point y such that y and r are on the same side of pq and $\angle pyq = \alpha$. The disk bounded by this circle is called α-$petal(pq)$. Similarly, consider the circle that goes through p, q and a third point z such that z and r are on the same side of pq and $\angle pzq = \gamma$. The disk bounded by this circle is called γ-$petal(pq)$. Then, the difference of the sets α-$petal(pq)$ and the *inside* of the γ-$petal(pq)$ is defined as $[\alpha, \gamma]$-$crescent(pq)$ (see Figure 4-2 (A)).

Let γ-$slab(pq)$ denote the region between the two lines one going through p, the other q and both making an angle γ to line segment pq. Similarly, let α-$wedge(pq)$ denote the region outside the two lines one going through p, the other q and both making an angle α to line segment pq. Then, $[\alpha, \gamma]$-$slice(pq)$ is defined as the intersection

of three regions; the γ-*slab*(pq), the α-*wedge*(pq), the $[\alpha,\gamma]$-*crescent*(pq). The $[\alpha,\gamma]$-*slice*(pq) must include another vertex, since otherwise there would be a bad triangle in the triangulation incident to p and q (see Figure 4-2 (B)).

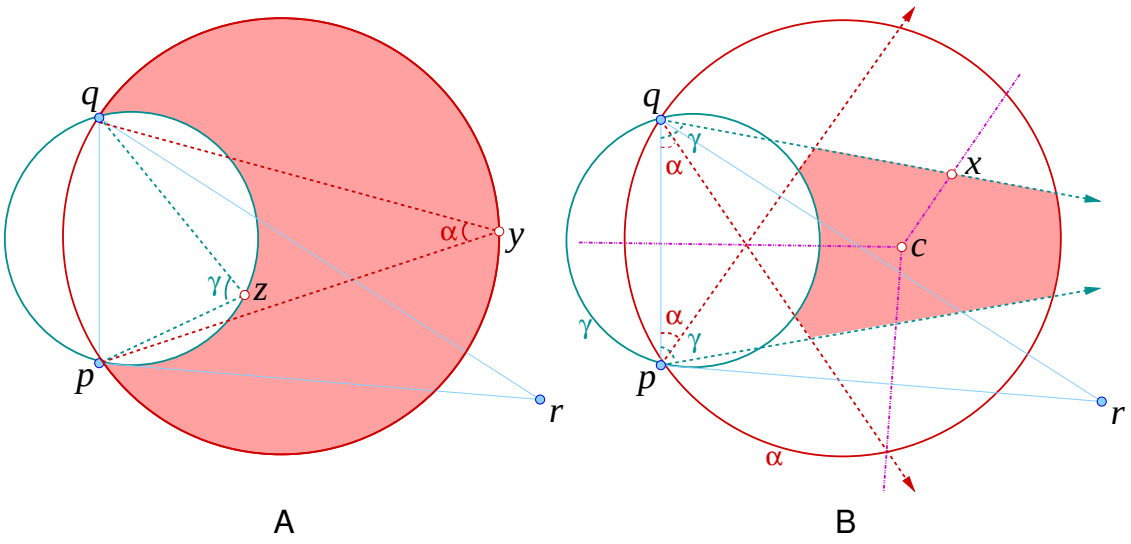


Figure 4-2. A) $[\alpha,\gamma]$-*crescent*(pq) is shown. B) Feasible region $[\alpha,\gamma]$-*slice*(pq) is the intersection of $[\alpha,\gamma]$-*crescent*(pq), α-*wedge*(pq), and γ-*slab*(pq) a point x inside this region that is furthest away from all existing vertices is chosen as a refinement point.

Hence, for point insertion procedure, we suggest to insert a Steiner point in the slice region of a bad triangle that is furthest away from all existing vertices. Note that, this point is not necessarily a Voronoi vertex or on a Voronoi edge. However, it is either on Voronoi skeleton or on the slice boundary. This is given in Lemma 3. Therefore, a simple local search on the Voronoi diagram help us to find that point inside the slice. Also observe that, while the α-wedge and the α-petal ensure the minimum angle constraint, the γ-slab and the γ-petal do the same for the maximum angle constraint.

The following lemma is similar to Lemma 1 in Chapter 3 where the search region is constructed considering only the α-*petal*(pq). The distinction here is due to the addition of the maximum angle constraint (imposed by γ-*slab*(pq) on the view angle and γ-*slab*(pq) on the side angles) and also the minimum angle constraint (imposed by γ-*slab*(pq)) on the side angles. Thanks to the addition of these constraints the region we

consider inserting Steiner points is more complex than before. Indeed unlike in the case of [53], the region is not necessarily convex.

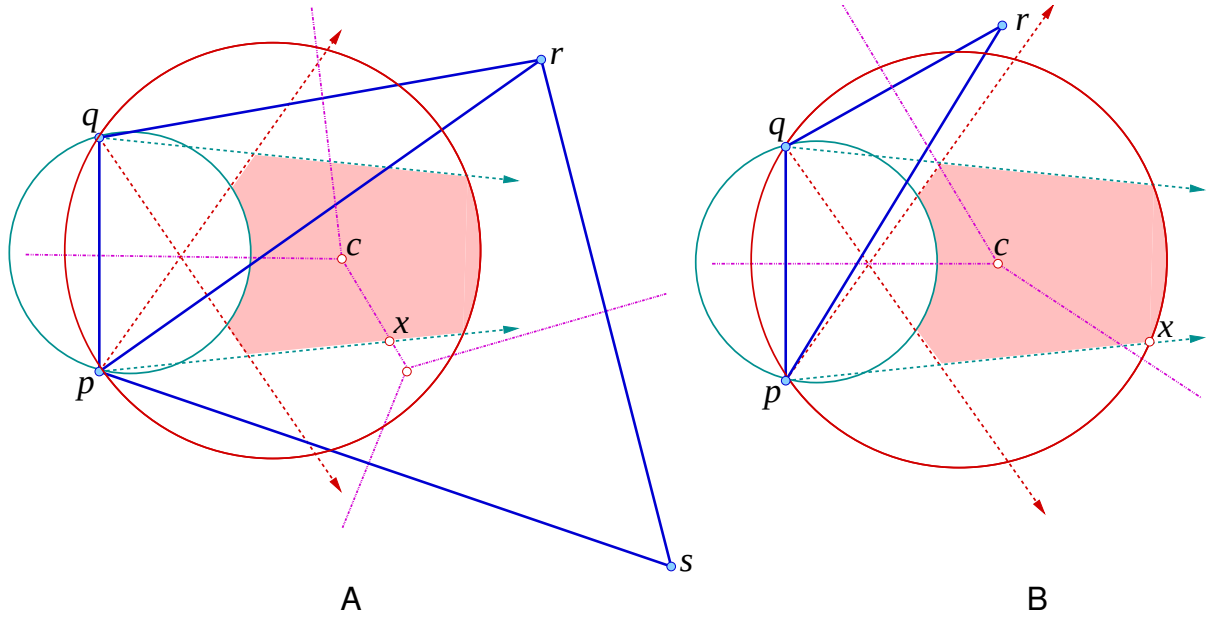


Figure 4-3. Additional types of Steiner points. A) Slab intersection case. B) Slab-petal intersection case.

Lemma 3. *During an iteration of the refinement process, a locally optimal (furthest from all existing vertices) Steiner point with respect to a slice region* $[\alpha, \gamma]$*-slice*(pq) *is on the Voronoi skeleton or on the boundary of the slice region.*

Proof. Consider a bad triangle pqr with the shortest edge pq. The $[\alpha, \gamma]$*-slice*(pq) is divided into regions bounded by lines (belong to α*-wedge*(pq) and γ*-slab*(pq)), arcs (belong to $\partial[\alpha, \gamma]$*-crescent*(pq)) and Voronoi edges. Let x be the locally optimal Steiner point limited to $[\alpha, \gamma]$*-slice*(pq). Suppose, for the sake of contradiction, that x is located neither on the Voronoi skeleton nor on the $\partial[\alpha, \gamma]$*-crescent*(pq). Since x is inside a Voronoi region and not on the boundary, x can be moved in any direction to go further away from the existing vertices. Hence, a point x' different than x always exists that is further from all vertices than x is, which contradicts our optimality assumption on x. □

In Chapter 3, we define a locally optimal Steiner point to be on a Voronoi edge or a Voronoi vertex and give four different types of Steiner points accordingly. Here, due to

the variety of the constraints and the complexity of the Steiner point insertion, additional classification is needed. Figure 4-3 demonstrates these new types of Steiner points as the intersection of a Voronoi edge and a slab line, and the intersection of a slab line and the petal.

4.3 Intersection of Slices and Free Vertex Relocation

Before attempting to remove a bad triangle using the above Steiner point insertion strategy, we consider relocating its free vertices (previously inserted Steiner points which are not part of the input). For each free vertex of a bad triangle, we compute the feasibility of moving it into a new location while making sure all triangles incident to it are good.

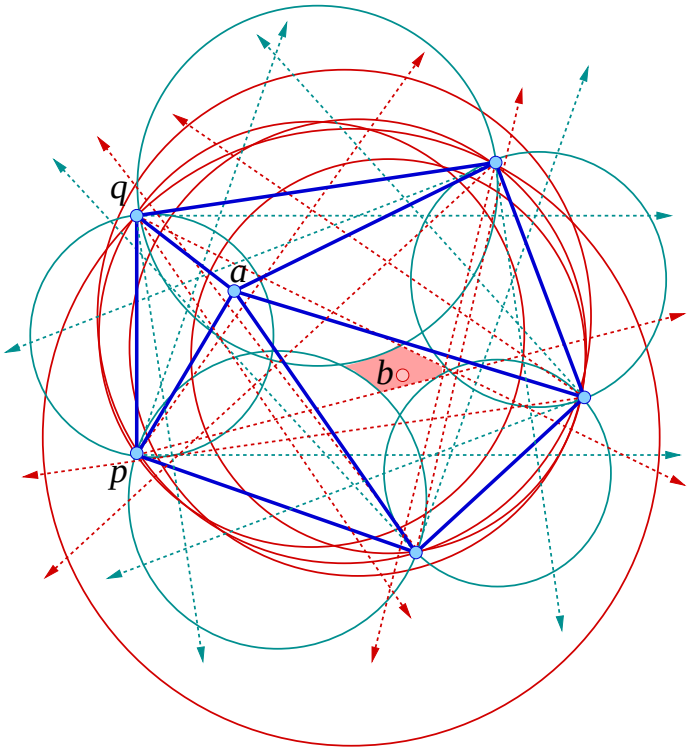


Figure 4-4. A bad triangle pqa is fixed by relocating one of its vertices a into b which is a point in the intersection (shaded) of the $[\alpha, \gamma]$-*slices* of the edges on the link of a.

The star of a vertex a consists of all triangles that contain a. The link of a, then, consists of all surrounding edges of triangles in the star that are not incident to a. A

relocation of a free vertex a is feasible only if intersection of the $[\alpha, \gamma]$-slices of the edges on the link is non-empty. See Figure 4-4.

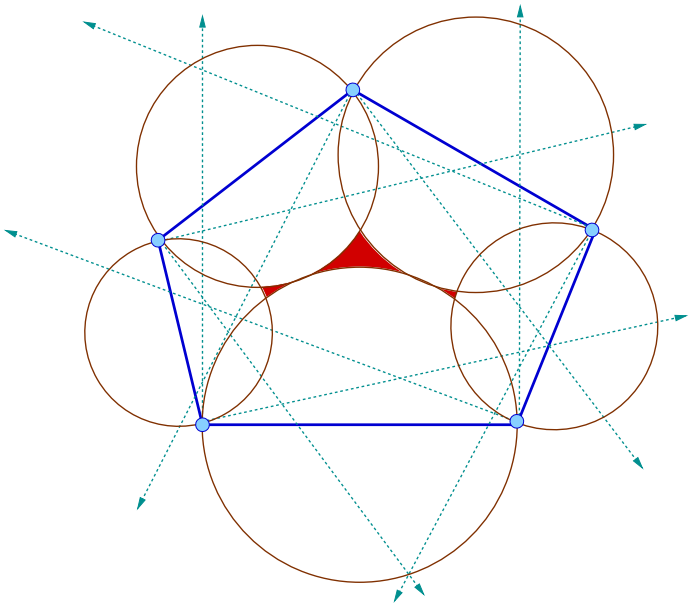


Figure 4-5. The intersection of the slices of link edges is not necessarily convex nor a single connected component. For clarity, we omit the α-wedge and α-petal constraints; after all a weak α constraint would result in the same feasible region.

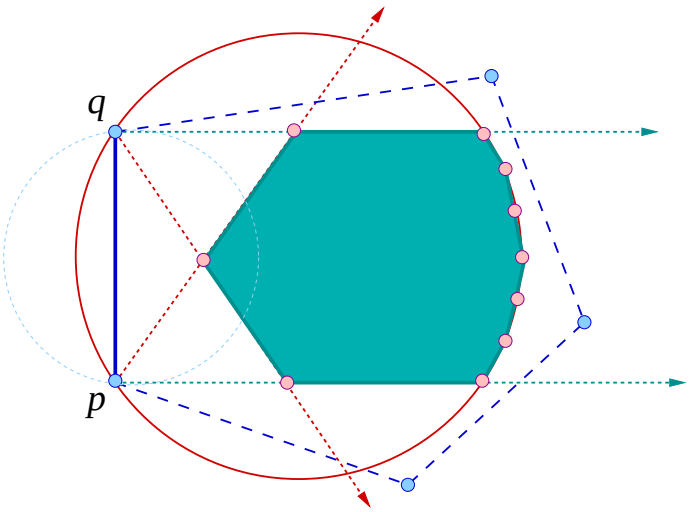


Figure 4-6. Approximate convex superset of a slice region. For efficient implementation, feasible intersection region for relocation is computed by using approximate slice regions.

Algorithm 4

```
Compute the Delaunay triangulation
  (DelTri) of the input
Let P denote the maintained point set
while ∃ a bad triangle pqr in DelTri(P)
  relocated := FALSE
  for each free vertex a of pqr
    for every edge xy on the link of a
      Compute α-wedge(xy)
      Compute γ-slab(xy)
      Compute [α, γ]-crescent(xy)
      Compute [α, γ]-slice(xy)
    endfor
    Compute 𝒦 = ⋂_{xy∈link(a)} [α, γ]-slice(xy)
    if 𝒦 ≠ ∅ and ∃b ∈ 𝒦 such that all triangles of
        star(b) in the DelTri(P ∪ {b} − {a}) are good
     then
      delete a; insert b; relocated:=TRUE; break;
  endfor
  if relocated == FALSE then
    insert a point x ∈ [α, γ]-slice(pq) that is furthest
      from all existing vertices
endwhile
```

As noted before, $[\alpha, \gamma]$-slice of an edge is usually not convex. As a result, the intersection of the slices of its link edges potentially consists of multiple sub-regions disconnected from each other. See Figure 4-5 which illustrates a feasible point relocation region with three independent components. Having curved boundary segments also adds difficulty to the complexity of computing feasible regions. In order to ease these complexities, we compute a convex approximate superset of the feasible region. The curved boundary segment due to each α-petal constraint is approximated with 6 linear segments. In addition, we postpone imposing the γ-petal constraints. See Figure 4-6. Once we compute a convex approximate superset of the feasible region, we do a sampling inside this region and check whether there is a sample point that makes all new triangles incident to it good. Algorithm 4 summarizes the integration of the refinement and the relocation steps.

4.4 Experiments

We tested an implementation of our algorithm on several complex data sets. Performance of the algorithm for various $[\alpha, \gamma]$ constraints are presented in Section 4.4.3. A comparative study against the existing refinement methods and smoothing methods are given in Sections 4.4.4 and 4.4.5. We also generate uniform triangulations without small and large angles with the additional maximum area constraint in Section 4.4.6.

4.4.1 Implementation

The implementation of the proposed method relies on an earlier implementation of the previous work in Chapter 3 [53], which integrates a point relocation step into the Delaunay refinement. The modifications, however, are substantial, as we deal with the complexities arising from the new constraints α-wedge, γ-petal and γ-slab. In the insertion step, we primarily enforce the α-petal and the γ-slab constraints, and do a lazy evaluation of the γ-petal and the α-wedge constraints. We perform a (graph) search on the Voronoi skeleton and the boundary of the slice region to locate a locally optimal Steiner point.

In the relocation step, we primarily enforce the α-petal, the α-wedge, and the γ-slab constraints, and do a lazy evaluation of the γ-petal constraint. Replacing the curved boundary segments of the α-petals with piecewise linear segments enables us to use the standard half-plane intersection algorithm for computing a convex approximate superset of the intersection of slices. Then, we perform a uniform sampling on this superset region in order to locate an actual feasible point for relocation.

4.4.2 Data Sets and Output Visualization

We have tested our method on complex two-dimensional domains of different types (point sets with various distributions, polygons with holes, and planar straight line graphs with small input angles). See the first column of Table 4-2. For the geometric domains

with small input angles e.g., Florida, Fish, Superior, we relied on the boundary handling strategy of the classical Delaunay refinement algorithms [99, 104].

Throughout this chapter, triangulations are visualized using the triangle coloring scheme described in Figure 4-7. This way, bad triangles whose angles are not in the interval $[\alpha, \gamma]$ stand out in the output visualization.

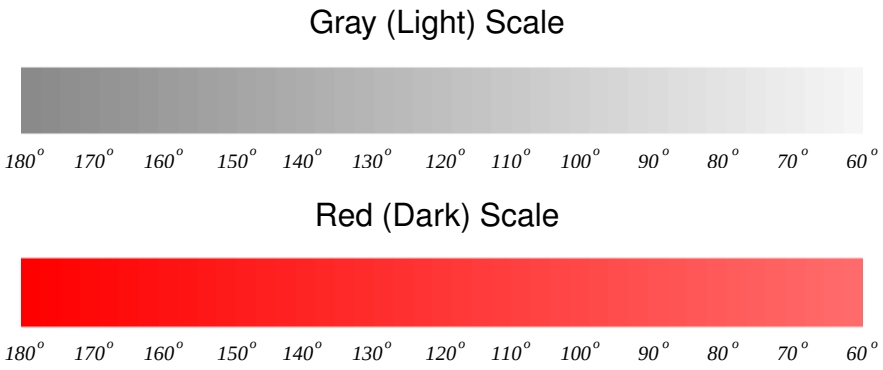


Figure 4-7. The gray (light) and the red (dark) scales are used for coloring the good and the bad triangles, respectively. In each case, the largest angle of the triangle determines the particular color.

4.4.3 Performance

We run our implementation for α varying from $0°$ to $40°$ and γ varying form $100°$ to $80°$ (each with increments of $1°$). The plots in Figure 4-8 illustrate that our algorithm is capable of handling almost all of these $800+$ $[\alpha, \gamma]$ combinations except when both α and γ are very strong constraints, e.g. $[\alpha, \gamma] = [40°, 80°]$. It is evident in this plot (and also in Table 4-2 and Figure 4-14) that as we enforce stronger minimum angle and/or maximum angle constraints the size of the output triangulation increases. This phenomenon is also shown in Figures 4-9 and 4-10. Upper part of the plots illustrates the relationship between the minimum angle constraint and the number of Steiner points inserted for a constant maximum angle constraint and vice versa. Concurrently, lower part of the plots reports the same for the running time, which reveals a parallel behavior. The plots for the other data sets are similar. Actual output triangulations and their histogram of angles

for complex domains are given in Figures 4-11 and 4-12. Here, different levels of quality triangulations are shown from good quality to premium.

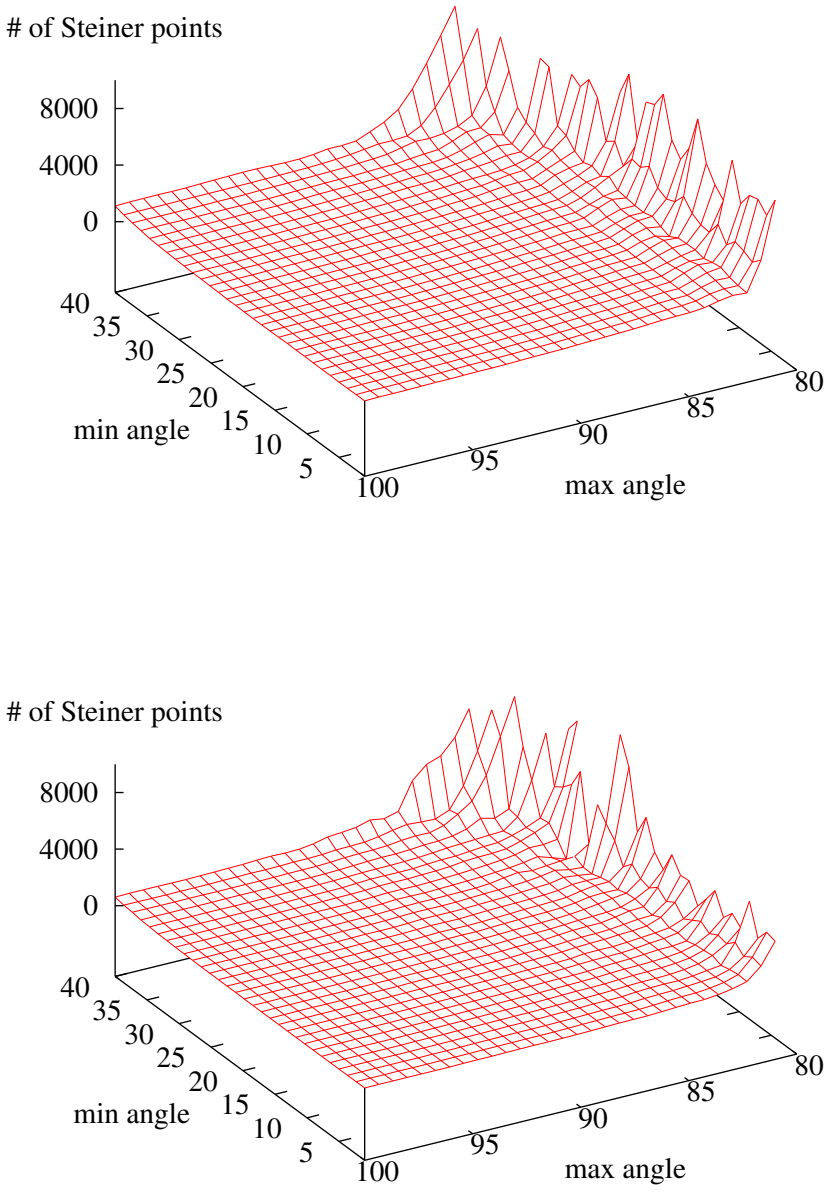


Figure 4-8. Plot of output size for varying α and γ for Lake Superior and Turkey data sets.

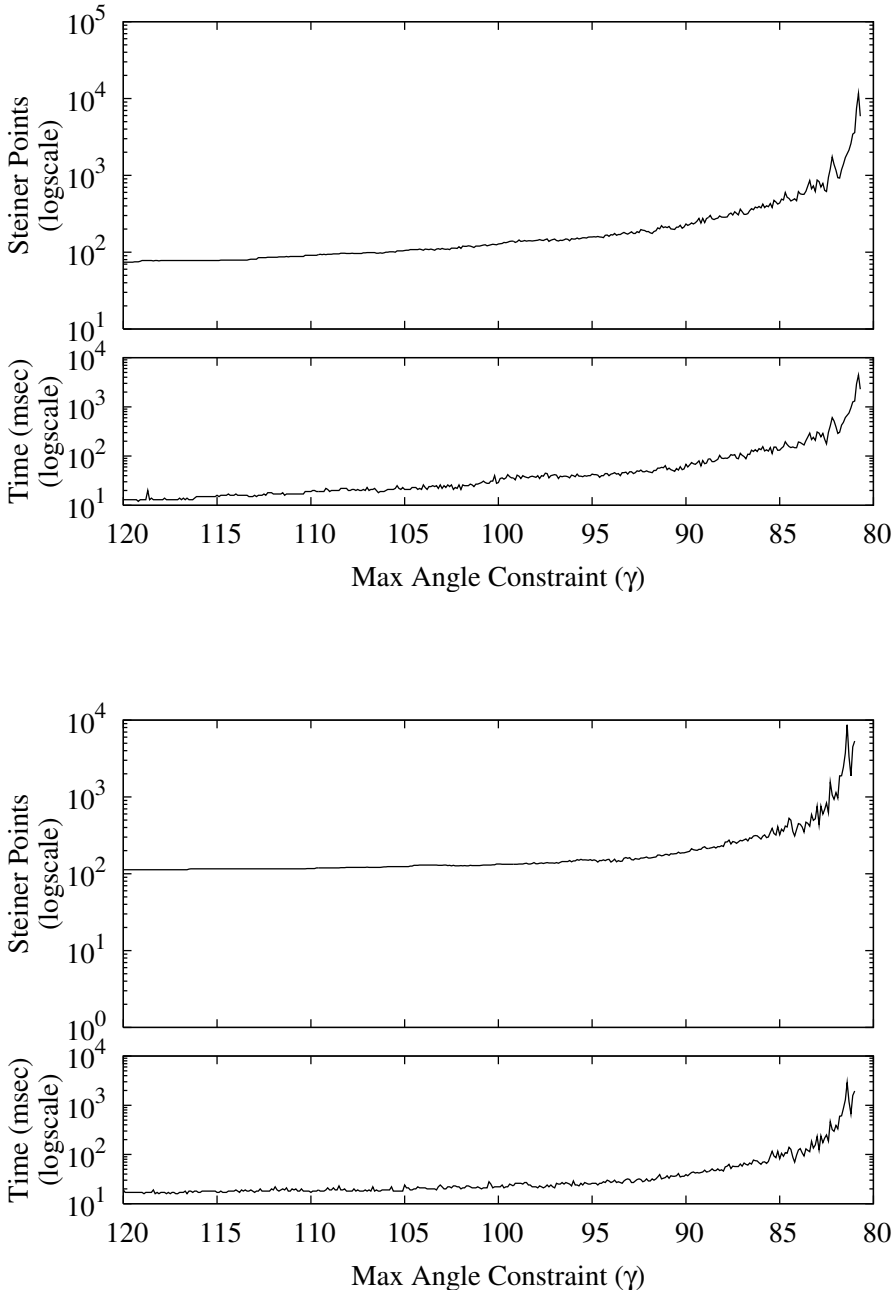


Figure 4-9. Plot of output size and time performance of the algorithm based on a fixed minimum angle constraint α and a variable maximum angle constraint γ for Turkey ($\alpha = 20°$) and Koala ($\alpha = 30°$) data sets.

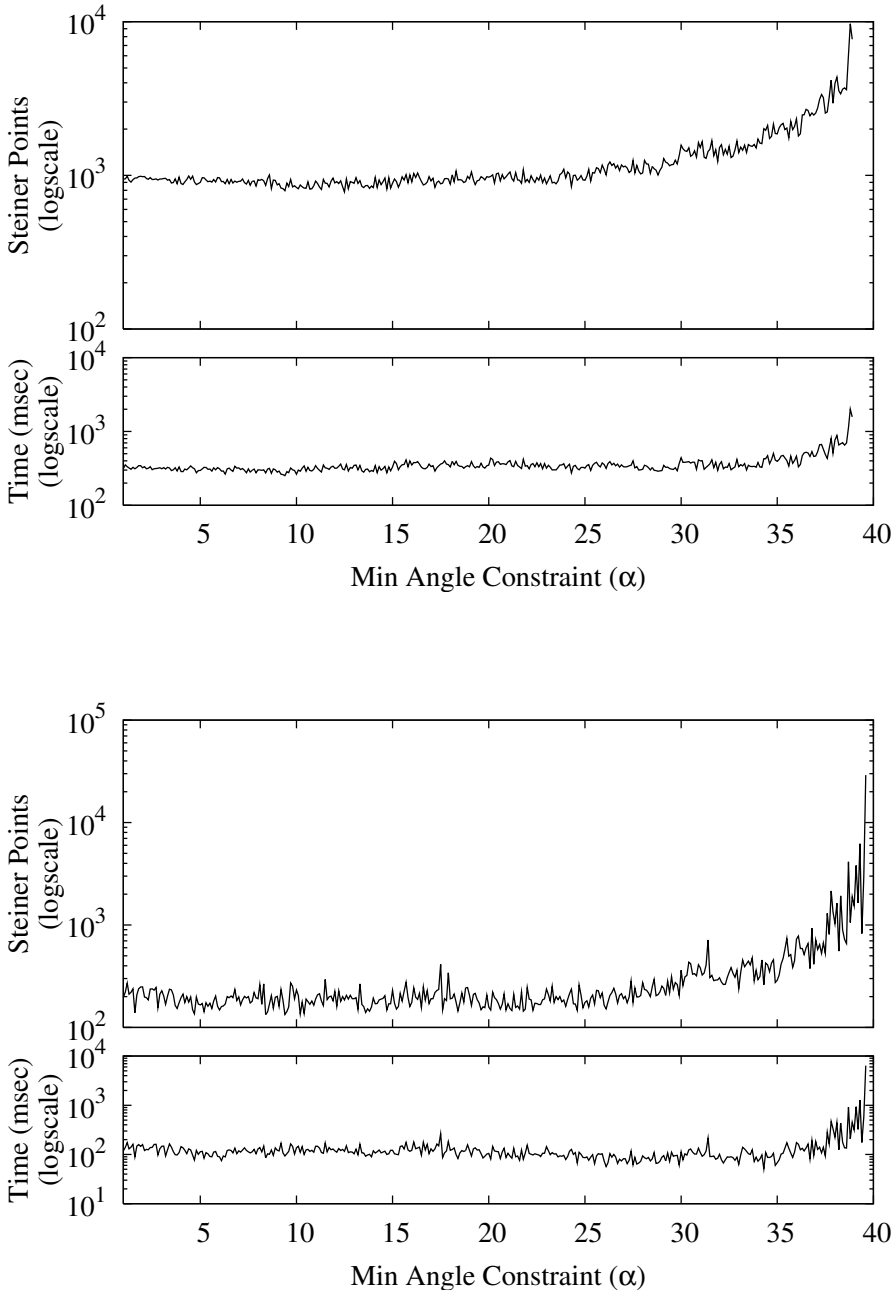


Figure 4-10. Plot of output size and time performance of the algorithm based on a fixed maximum angle constraint γ and a variable minimum angle constraint α for California ($\gamma = 85°$) and Boeing ($\gamma = 85°$) data sets.

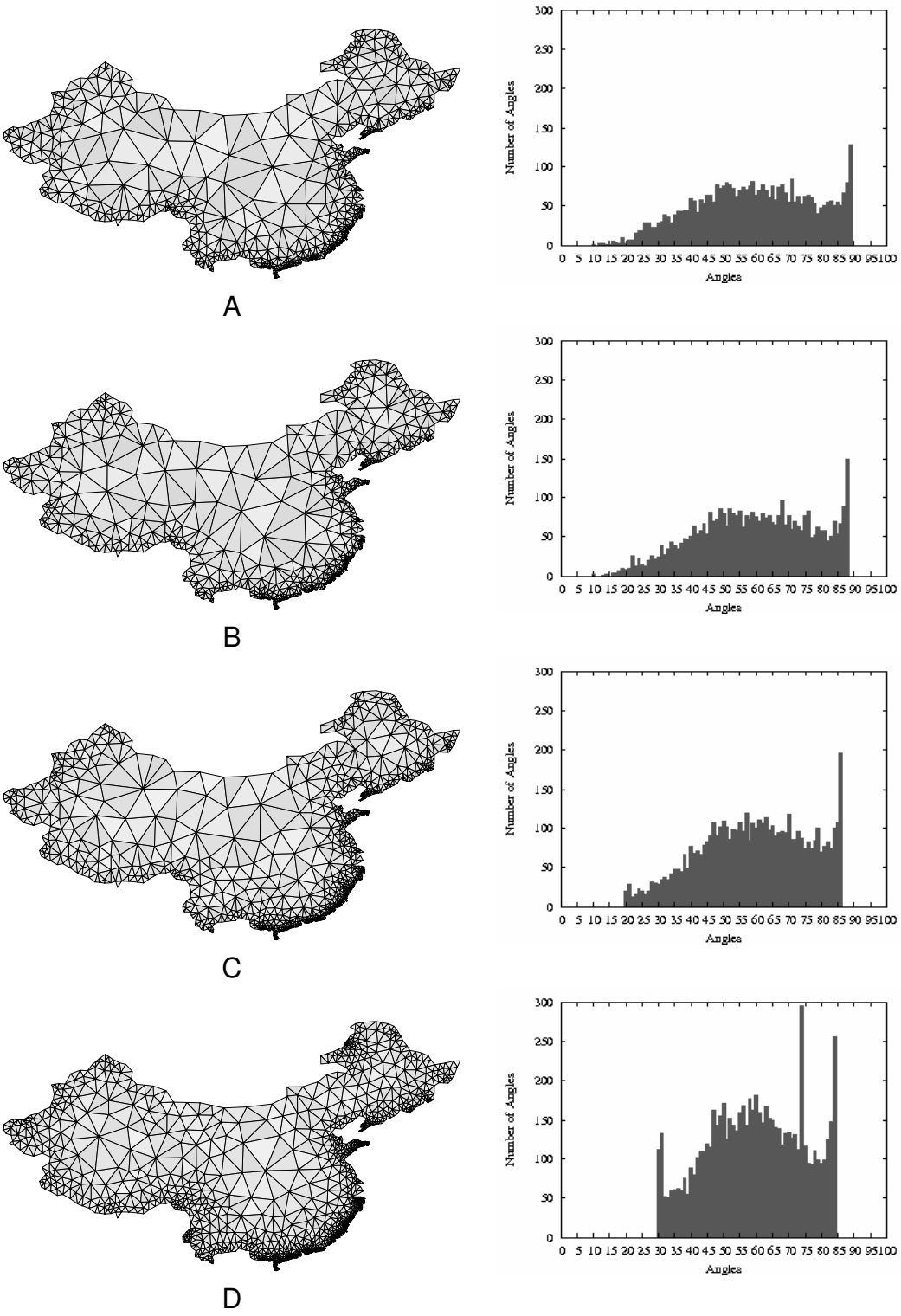


Figure 4-11. Output triangulations and their angle histograms of China data set for various minimum and maximum angle constraints. A) $[\alpha, \gamma] = [1°, 90°]$, 849 points. B) $[\alpha, \gamma] = [10°, 89°]$, 889 points. C) $[\alpha, \gamma] = [20°, 87°]$, 1073 points. D) $[\alpha, \gamma] = [30°, 85°]$, 1409 points.

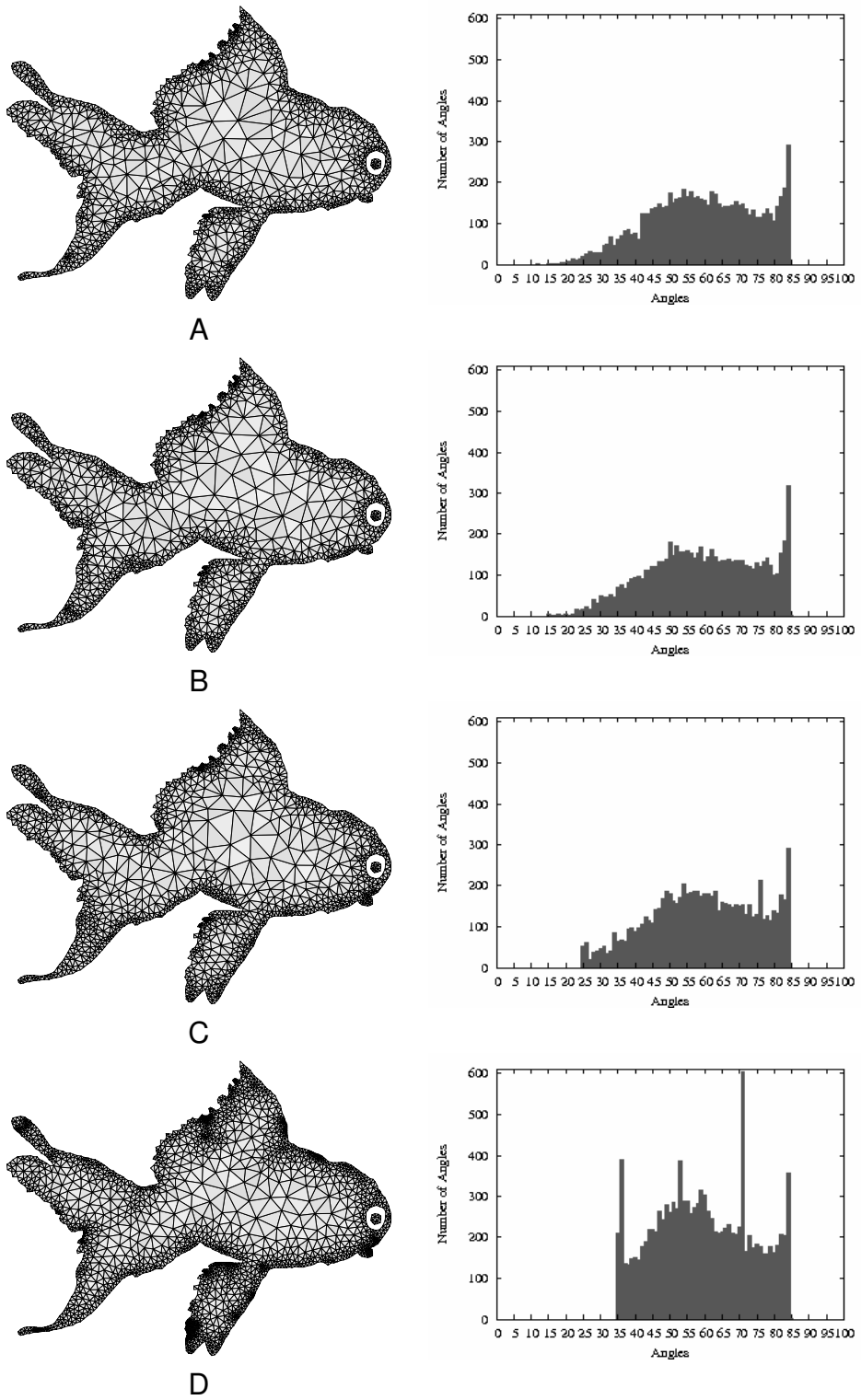


Figure 4-12. Output triangulations and their angle histograms of goldfish data set for various minimum and maximum angle constraints. A) $[\alpha, \gamma] = [5°, 85°]$, 1497 points. B) $[\alpha, \gamma] = [15°, 85°]$, 1454 points. C) $[\alpha, \gamma] = [25°, 85°]$, 1584 points. D) $[\alpha, \gamma] = [35°, 85°]$, 2249 points.

Table 4-2. Performance of the algorithm for four different $[\alpha, \gamma]$ ranges on 15 different data sets. (Continued on the next page.)

	Input			Output		Time
Data set	#Points	#Segments	#Holes	[α,γ]	#Points	(msec)
Cyprus	448	447	0	$[30°, 90°]$	1540	211
				$[35°, 83°]$	8918	1810
				$[40°, 86°]$	9755	1270
				$[25°, 81°]$	16678	6001
Turkey	216	216	0	$[30°, 90°]$	571	62
				$[35°, 83°]$	1796	341
				$[40°, 86°]$	4496	608
				$[25°, 81°]$	8973	3412
Italy	216	216	0	$[30°, 90°]$	489	142
				$[35°, 83°]$	1293	225
				$[40°, 86°]$	4704	1256
				$[25°, 81°]$	3269	1395
Iraq	192	192	0	$[30°, 90°]$	466	249
				$[35°, 83°]$	2345	472
				$[40°, 86°]$	3475	621
				$[25°, 81°]$	7217	2903
Florida	304	304	0	$[30°, 90°]$	895	139
				$[35°, 83°]$	3645	705
				$[40°, 86°]$	4084	516
				$[25°, 81°]$	8901	3233
Superior	522	522	7	$[30°, 90°]$	1250	157
				$[35°, 83°]$	3965	751
				$[40°, 86°]$	5308	661
				$[25°, 81°]$	7532	2701
Fish	92	92	1	$[30°, 90°]$	205	258
				$[35°, 83°]$	528	99
				$[40°, 86°]$	2399	356
				$[25°, 81°]$	2845	1043
Dolphin	200	200	1	$[30°, 90°]$	456	64
				$[35°, 83°]$	1339	283
				$[40°, 86°]$	1430	166
				$[25°, 81°]$	3165	1130
Alligator	1100	1100	3	$[30°, 90°]$	4205	618
				$[35°, 83°]$	15602	3086
				$[40°, 86°]$	21425	2669
				$[25°, 81°]$	47678	17960
Hexagon	8	6	0	$[30°, 90°]$	27	5
				$[35°, 83°]$	46	7
				$[40°, 86°]$	53	6
				$[25°, 81°]$	27	9

Table 4-2: Continued.

	Input			Output		Time
Data set	#Points	#Segments	#Holes	$[\alpha,\gamma]$	#Points	(msec)
Boeing	30	30	1	$[30°, 90°]$	158	51
				$[35°, 83°]$	640	152
				$[40°, 86°]$	1053	145
				$[25°, 81°]$	2279	1035
Boxed Pair	6	4	0	$[30°, 90°]$	57	34
				$[35°, 83°]$	431	108
				$[40°, 86°]$	219	31
				$[25°, 81°]$	1364	730
Kuzmin	2000	0	0	$[30°, 90°]$	6269	1322
				$[35°, 83°]$	19086	4247
				$[40°, 86°]$	29086	4185
				$[25°, 81°]$	61661	23922
Plate	65	64	5	$[30°, 90°]$	249	45
				$[35°, 83°]$	2718	616
				$[40°, 86°]$	3664	501
				$[25°, 81°]$	4910	1865
Random	1000	0	0	$[30°, 90°]$	3426	761
				$[35°, 83°]$	12062	2807
				$[40°, 86°]$	15410	2345
				$[25°, 81°]$	31122	12108

Table 4-2 presents the performance (time and output size) of our algorithm on 15 data sets for four different combinations of $[\alpha, \gamma]$ values. It should be noted that our algorithm handles the $[\alpha, \gamma] = [30°, 90°]$ constraint with great ease, resulting in a fast computation of small size high quality non-obtuse triangulations. Time performance is still reasonable for somewhat stronger constraints, e.g. $[\alpha, \gamma] = [35°, 83°]$. However, pushing either one of these constraints further stronger requires a slight relaxation of the other constraint in order to assure the termination of the algorithm. For instance, the interval $[25°, 81°]$ (very strong maximum angle constraint) and the interval $[40°, 86°]$ (very strong minimum angle constraint) are two $[\alpha, \gamma]$ constraints for which our algorithm is shown to work on all the experimented data sets. The results presented here should be taken as a testament to the broad capabilities of our method and the software.

Table 4-3. Comparison of our method with the previous Delaunay refinement methods (classical Delaunay refinement method [28, 45, 99, 104, 118] and LOSP method [53]) in terms of the output size (number of points) and the angle range $[\alpha, \gamma]$.

Data set	Classical Del. Ref. $[\alpha, \gamma]$	#Pts	$\gamma - \alpha$	LOSP Method $[\alpha, \gamma]$	#Pts	$\gamma - \alpha$	Our Method $[\alpha, \gamma]$	#Pts	$\gamma - \alpha$
Turkey	[35.0°, 110.0°]	1247	75	[41.0°, 97.0°]	1317	56	[35.0°, 84.0°]	1168	49
							[38.0°, 87.0°]	1283	49
							[38.0°, 89.0°]	891	51
							[41.0°, 93.0°]	1476	52
Florida	[34.7°, 109.0°]	1422	74	[40.0°, 100°]	1326	60	[35.0°, 84.0°]	1168	49
							[38.0°, 86.0°]	1320	48
							[38.0°, 88.0°]	1056	50
							[40.0°, 90.0°]	1433	50
Superior	[35.0°, 109.0°]	2428	74	[41.5°, 97.0°]	2339	55	[35.0°, 85.0°]	2246	50
							[38.0°, 86.0°]	2382	48
							[38.0°, 88.0°]	1862	50
							[41.5°, 94.0°]	2562	52
Boeing	[35.0°, 108.0°]	349	73	[41.0°, 97.0°]	379	56	[35.0°, 85.0°]	400	50
							[38.0°, 86.0°]	381	48
							[38.0°, 89.0°]	321	51
							[41.0°, 91.0°]	316	50
Boxed Pair	[35.4°, 109.0°]	227	73	[42.0°, 95.0°]	178	53	[35.4°, 83.0°]	250	48
							[38.0°, 85.0°]	232	47
							[38.0°, 88.0°]	111	50
							[42.0°, 91.0°]	168	49
Kuzmin	[34.7°, 110.5°]	12548	76	[40.5°, 99.0°]	11824	58	[35.0°, 86.0°]	11002	51
							[38.0°, 87.0°]	12476	49
							[38.0°, 89.0°]	10380	51
							[40.5°, 95.0°]	12086	54
Plate	[35.5°, 108.5°]	634	73	[41.0°, 98.0°]	598	57	[35.5°, 85.0°]	653	50
							[38.0°, 86.0°]	663	48
							[38.0°, 88.0°]	465	50
							[41.0°, 91.0°]	565	50
Random	[34.9°, 110.0°]	7122	75	[41.0°, 98.0°]	6992	57	[35.0°, 85.0°]	6654	50
							[38.0°, 87.0°]	7080	49
							[38.0°, 89.0°]	5798	51
							[41.0°, 93.0°]	7335	51
Fish	[35.7°, 107.0°]	547	71	[41.5°, 97.0°]	498	55	[35.0°, 83.0°]	537	48
							[38.0°, 85.0°]	636	47
							[38.0°, 87.0°]	348	49
							[41.5°, 94.0°]	581	52
Iraq	[35.0°, 110.0°]	1256	75	[41.5°, 97.0°]	1117	55	[35.0°, 84.0°]	1118	49
							[38.0°, 86.0°]	1054	48
							[38.0°, 88.0°]	825	50
							[41.5°, 92.0°]	1291	50

4.4.4 Comparison: Previous Delaunay Refinement Algorithms

We conducted several experiments to compare our method with previous Delaunay refinement methods. Table 4-3 compares classical Delaunay refinement method [28, 45, 99, 104, 118], LOSP method [53] and our method in terms of maximum and minimum angle as well as output sizes of the triangulations. Our experiments show that classical Delaunay refinement method gives maximum angles as low as $110°$, whereas LOSP method reduce this further to $98°$. Although, these algorithms can provide large smallest angles, this does not yield to small largest angles. On the other hand, our method, for nearly the same number of output points, can achieve minimum angles as high as $35°$ while keeping the maximum angles as low as $85°$. For larger α values, without sacrificing from the output size, we can reach smaller γ values as compared to previous Delaunay refinement-based methods. Significant difference in range values indicates the strength of our method, such that classical Delaunay refinement method and LOSP method provides interval ranges $\gamma - \alpha$ around 75 and 57, respectively, whereas this value is around 49 for our method.

Figures 4-13 and 4-14 illustrate the output of the classical Delaunay refinement and our method, respectively, for varying $[\alpha, \gamma]$ constraints. Note that classical Delaunay refinement takes only the α constraint as input. In Figure 4-13, there are indeed only three different triangulations one for each of $\alpha = 11°, 22°, 33°$. For comparison, we colored the bad triangles with respect to the corresponding γ constraints imposed in the implementation of the proposed method. Out of the twelve different combinations of $[\alpha, \gamma]$ constraints, the classical Delaunay refinement method gives an output acceptable by only one combination ($[33°, 115°]$), where this is mainly because the upper bound for γ is actually in the trivial large angle bound, $180° - 2 * 33° = 114°$. On the other hand, our method successfully generates quality triangulations (Figure 4-14) for all corresponding $[\alpha, \gamma]$ values. (See also Figure 4-17.)

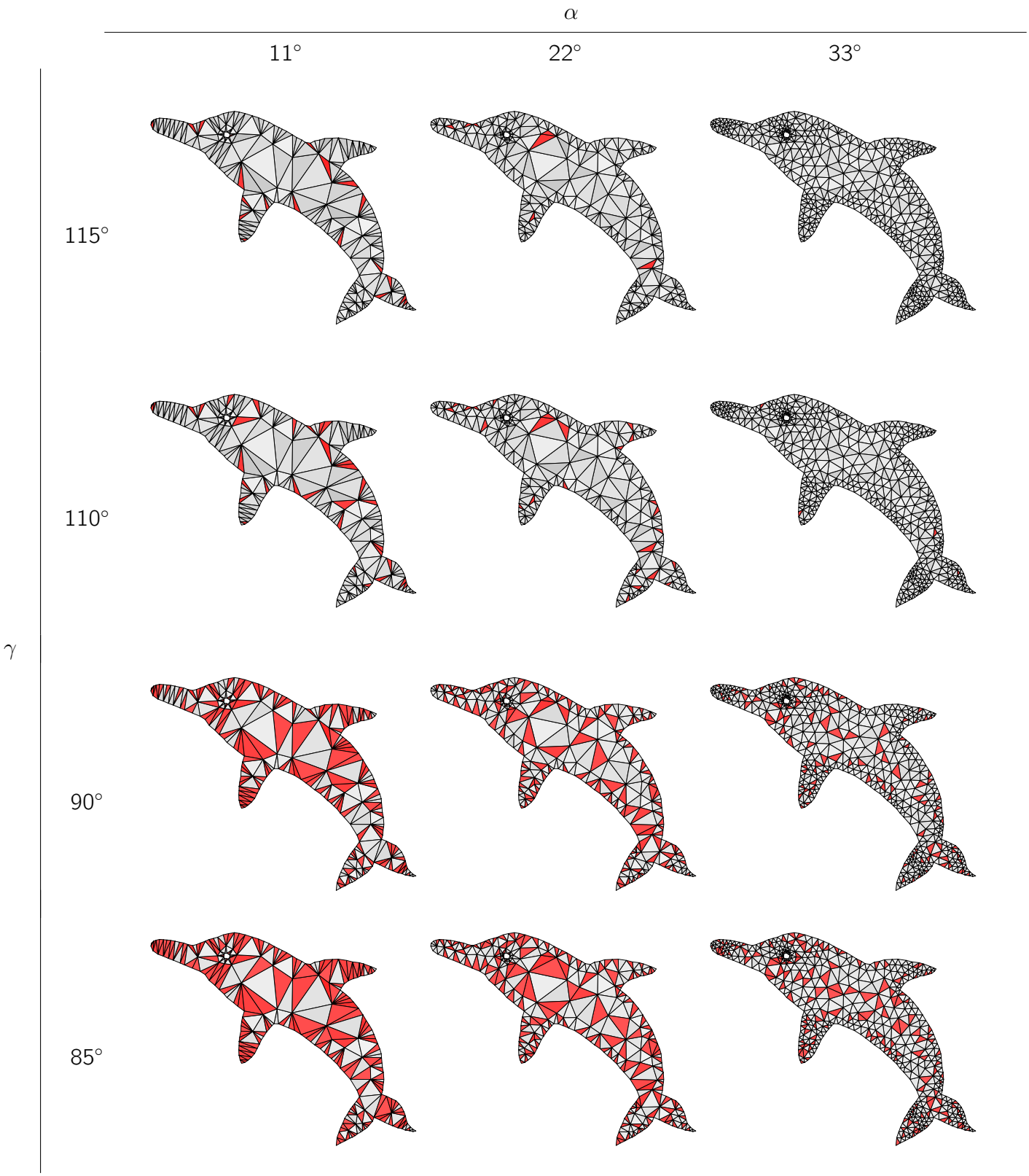


Figure 4-13. Output of the classical Delaunay refinement method for $\alpha = 11°$, $22°$, and $33°$. Triangles are marked according to four different γ constraints.

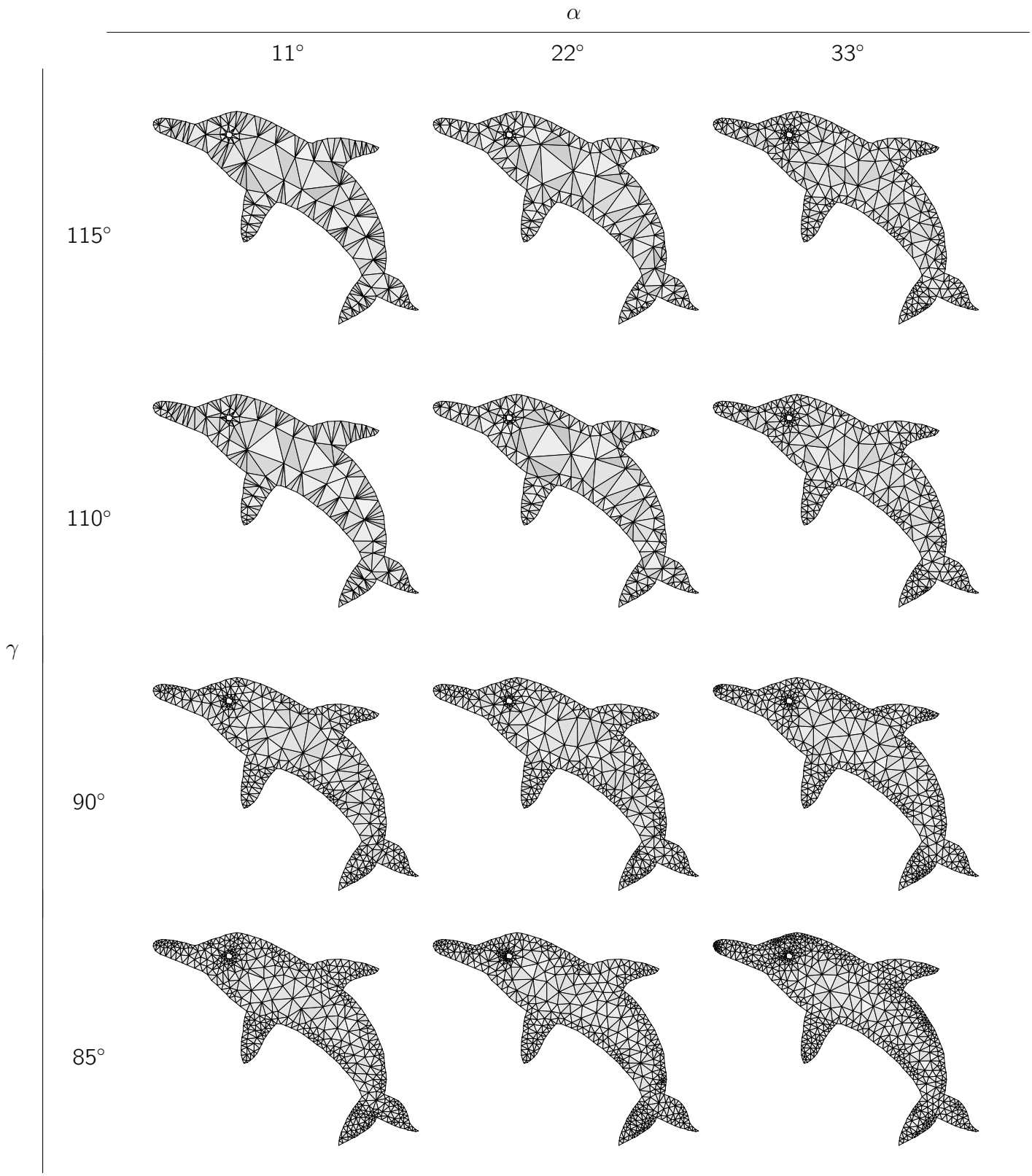


Figure 4-14. Output of our method based on maximum and minimum angle constraints $[\alpha, \gamma]$.

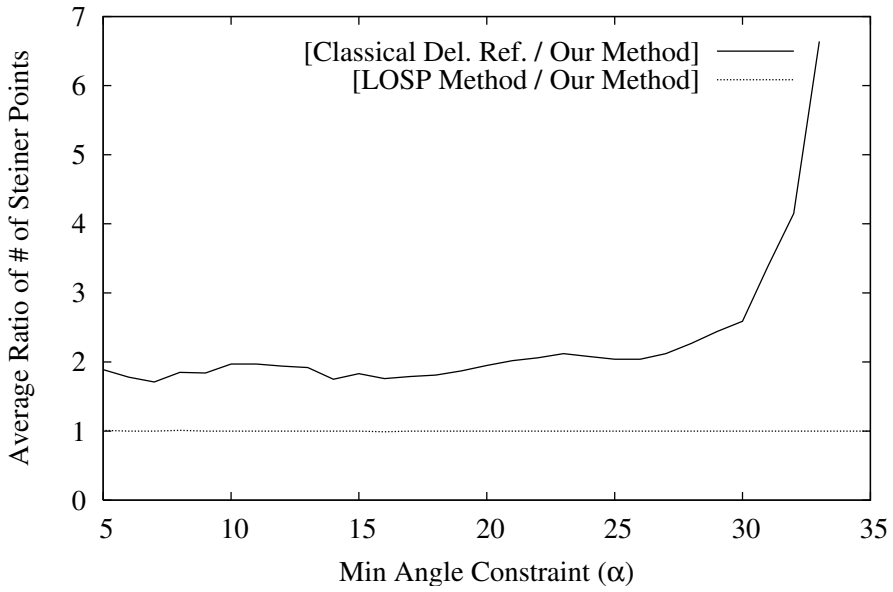


Figure 4-15. Size comparison of all data sets in Table 4-2 for $[\alpha, \gamma] = [30°, 120°]$ interval by applying classical Delaunay refinement method, LOSP method and our method.

In Figure 4-15, we plot the ratio of the number of Steiner points inserted by the Delaunay refinement method and our method averaged over 15 data sets. For this we naturally enforced an $[\alpha, \gamma]$ interval which can be achieved by the classical Delaunay refinement and LOSP methods. The output size of our method is roughly the same as the LOSP method, whereas we generate significantly smaller triangulations than classical Delaunay refinement method. This is in agreement with the observation made in Table 4-3 that our method generates better quality triangulations using nearly the same number of points. Also, Figure 4-16 gives an output comparison example for a reasonably high α and a trivial $\gamma = \pi - 2\alpha$ constraint.

4.4.5 Comparison: Smoothing Methods

We compare our method against the WCT method of [120] which is designed to iteratively smooth an input triangulation into an acute triangulation. For this experiment, we provide the output of a classical Delaunay refinement method (Figure 4-17 (A)), as an input to the WCT method. As seen in Figure 4-17 (C), WCT method improves

the largest angle by about 5° but in the meantime deteriorates the smallest angle by about 8°. It should be noted that while WCT smoothing is designed to compute acute triangulations, it generally fails to do so for complex geometric inputs unless the input triangulation is already acute or almost acute [120]. We also include an output of the LOSP method for comparison in Figure 4-17 (B). Using nearly the same number of triangles as the previous methods, our method computes triangulations with much better quality keeping all angles in $[35^\circ, 85^\circ]$ a significant improvement over the WCT's $[25^\circ, 97^\circ]$. Moreover, the proposed method is more than an order of magnitude faster than the WCT method and has comparable speed to the LOSP and the classical Delaunay refinement methods. Comparative experiments on other data sets reveal similar results.

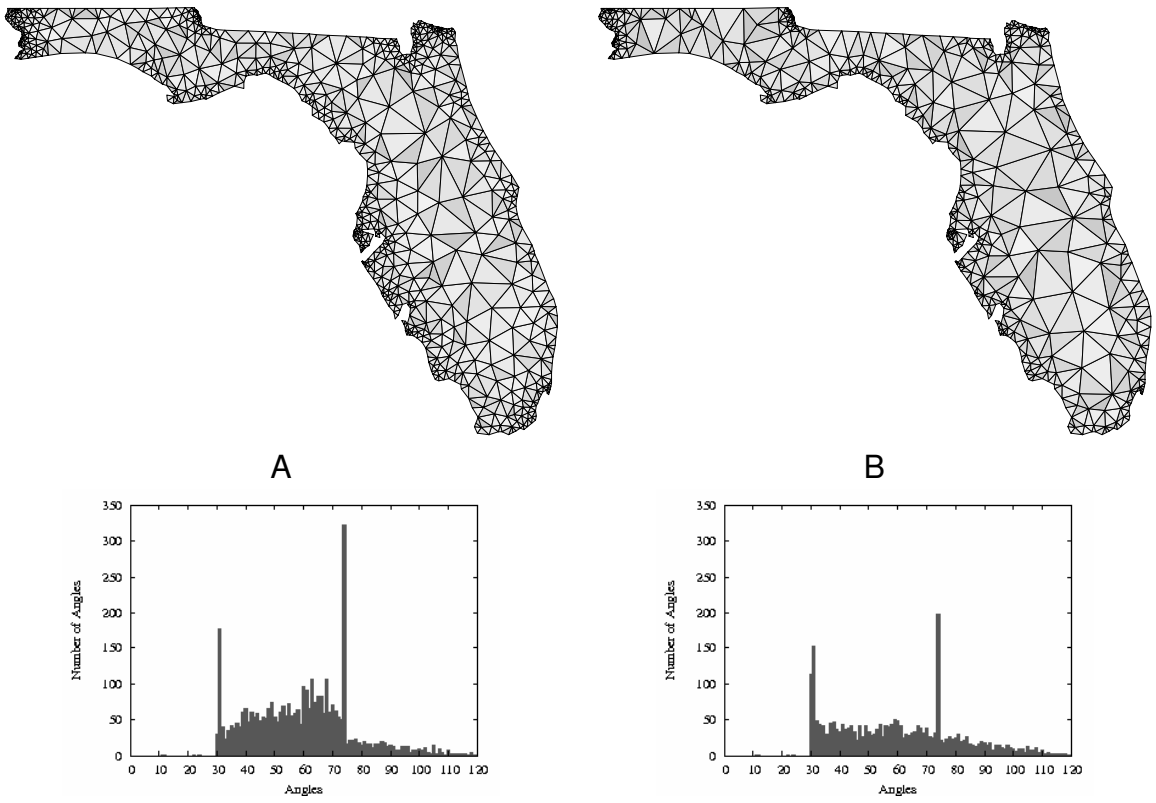


Figure 4-16. Comparing output sizes and angle histograms of previous classical Delaunay refinement algorithm and ours for the same $[\alpha, \gamma] = [30^\circ, 120^\circ]$ interval. A) Classical Del. Ref. [99, 118], 759 points. B) Classical Del. Ref. [99, 118], 588 points.

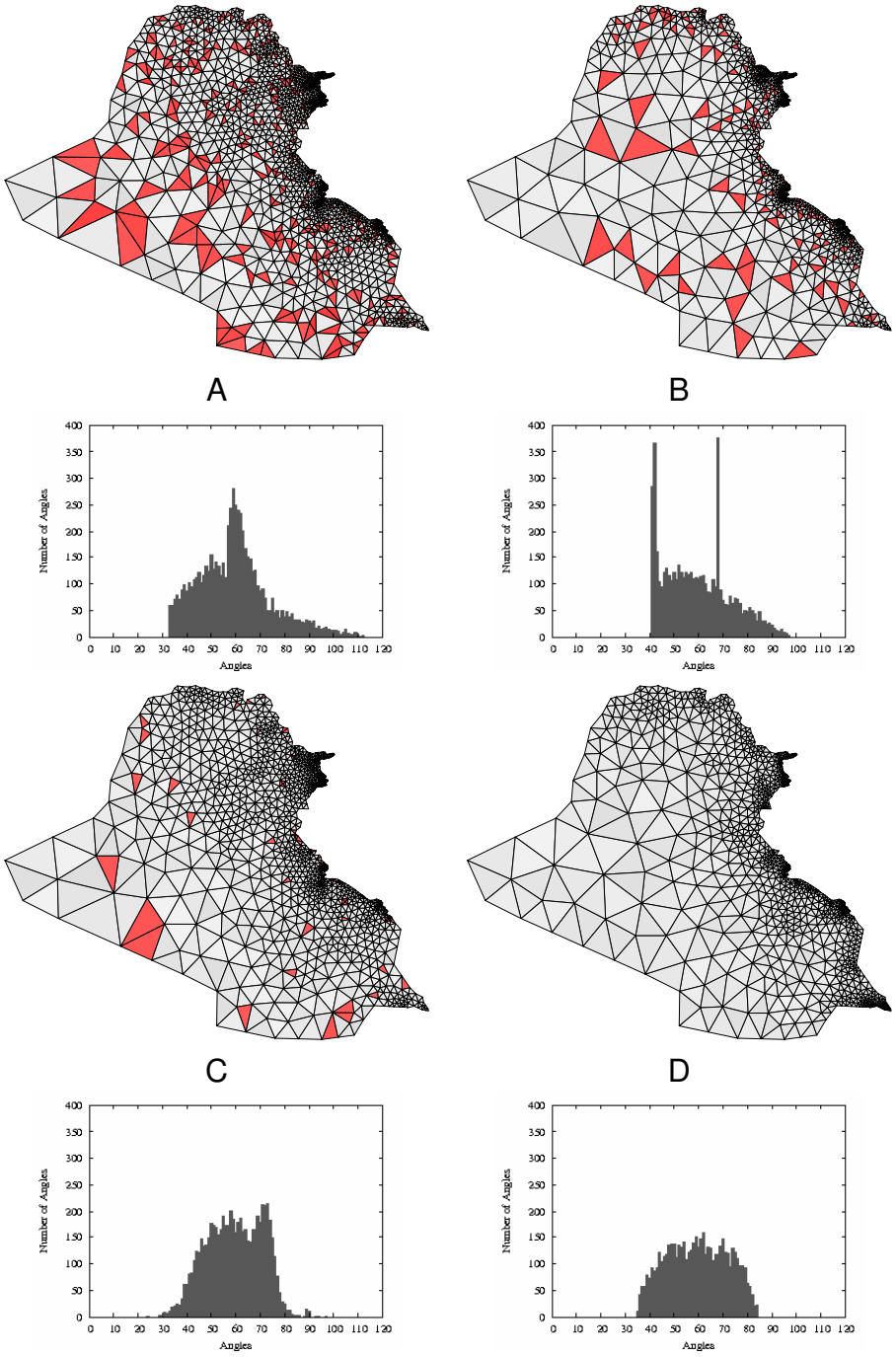


Figure 4-17. Iraq map. Output of (A) the classical Delaunay refinement for $\alpha = 33°$, (B) the LOSP method for $\alpha = 41°$, (C) the WCT method where (A) is used as input, and (D) the proposed method for $[\alpha, \gamma] = [35°, 85°]$. Triangles with an angle outside the interval $[\alpha, \gamma] = [35°, 85°]$ are marked as bad. A) Classical Del. Ref. [99, 118] (Initial), $[\alpha, \gamma] = [33.021°, 112.7°]$, 1155 points, 66574 msec. B) LOSP Method [53], $[\alpha, \gamma] = [41°, 97.7°]$, 1007 points, 185 msec. C) WCT Method [120], $[\alpha, \gamma] = [24.918°, 97.711°]$, 1155 points, 65 msec. D) Our Method $[\alpha, \gamma] = [35.15°, 85°]$, 971 points, 38 msec.

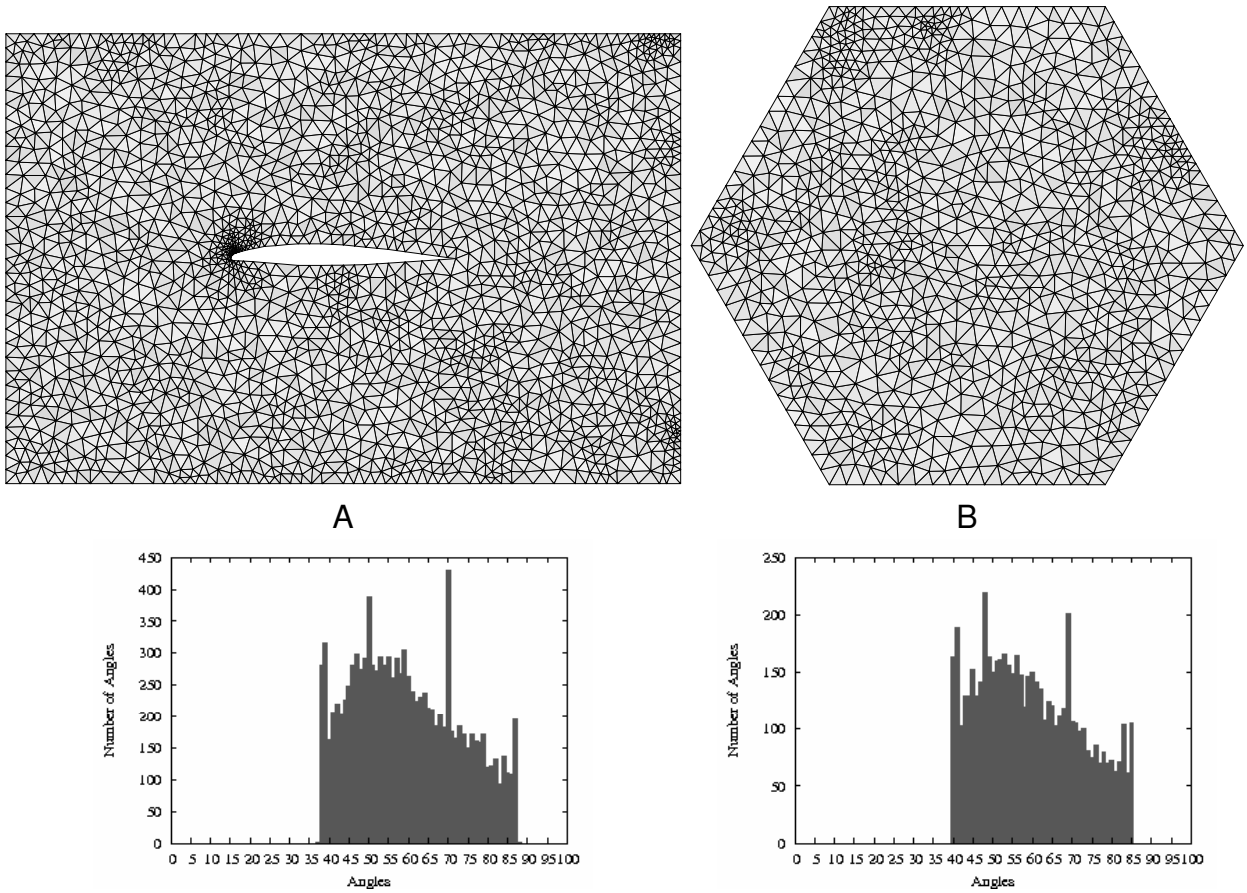


Figure 4-18. Premium quality output triangulations for Boeing and hexagon input domains with the additional maximum area constraint. A) $[\alpha, \gamma] = [38°, 88°]$, max area = 0.003, shortest edge = 0.0036. B) $[\alpha, \gamma] = [40°, 86°]$, max area = 0.05, shortest edge = 4.

4.4.6 Results on Uniform Triangulations

In addition to minimum and maximum angle constraints, we also applied maximum triangle area constraint. For applications requiring uniform triangulations, usually maximum triangle area is used as an element size constraint. Given in Figure 4-18 and 4-19, we give output triangulations for different data sets for various $[\alpha, \gamma]$ intervals and maximum area constraints. While Figure 4-18 shows the strength of our method to produce uniform triangulations without small and large angles for significantly good intervals, Figure 4-19 compares uniform triangulations for different size constraints for the same angle ranges.

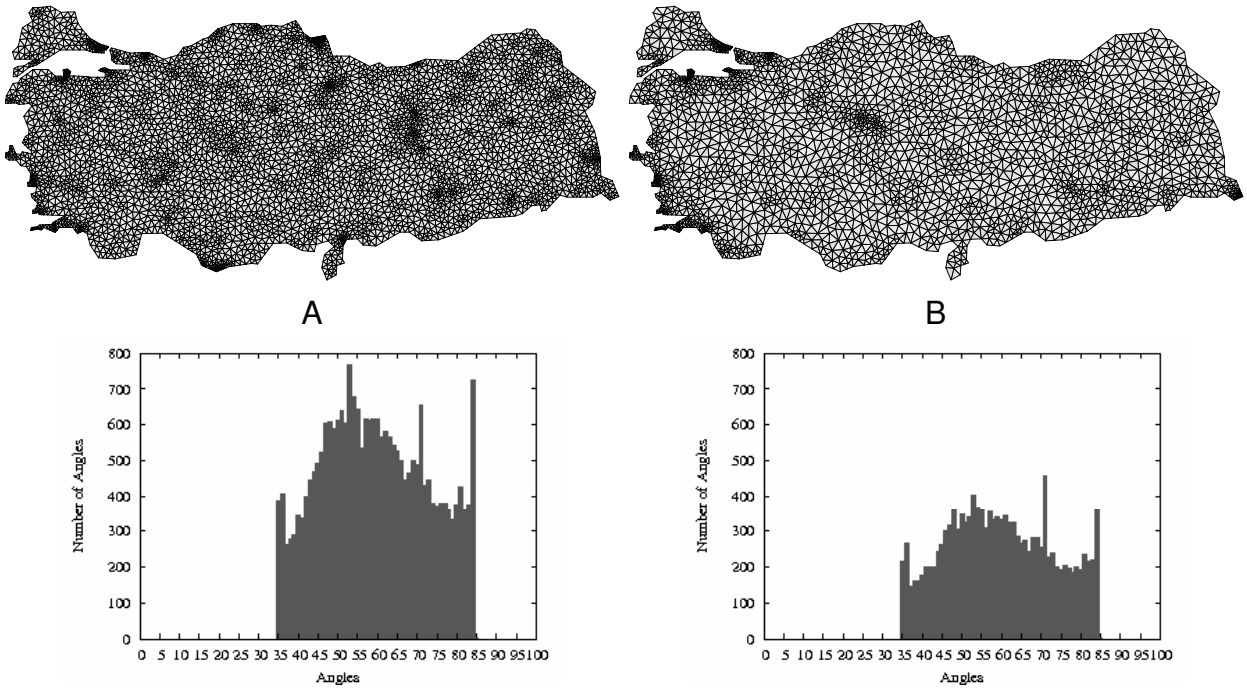


Figure 4-19. Premium quality output triangulations for Turkey data set with different maximum area constraints. A) $[\alpha, \gamma] = [35°, 85°]$, max area $= 100$, shortest edge $= 3.6$. B) $[\alpha, \gamma] = [35°, 85°]$, max area $= 200$, shortest edge $= 3.6$.

CHAPTER 5
DISCUSSIONS

5.1 aCute Software

We combine our ideas given in this thesis and provide a University of Florida licensed software called aCute for generating premium quality triangulations. To our knowledge, this is the first software generating such triangulations. We believe researchers and many engineering applications will benefit from using our software.

5.1.1 Software Website

We provide a website for the users, where they can reach the important information related to our software [52]. Website also includes the descriptions of our ideas and give references to our published work. Moreover, a detailed comparison with Triangle, which is one of the most popular triangulation software, is given. We provide sample animations of our algorithms to describe the process and point out important details/challenges.

5.1.2 Download and Usage

aCute software relies on the data structures and basic Delaunay triangulation algorithms that have been implemented in Triangle software. Mainly, this new software replaces the original circumcenter insertion strategy with our new point insertion and local relocation techniques. That is why, the implementation uses Triangle software as basis and add the necessary computations. Hence, users also need Triangle software for using aCute software. All the necessary instructions are given in the documentations. This also facilitates users to benefit from several different options already given in Triangle software. For no small angle triangulations, the user need to specify a threshold value for the smallest angle as usual, whereas for no small no large angle triangulations an additional threshold value for largest angle is needed as well.

5.1.3 Comparison: Differences and Similarities with Triangle Software

A detailed comparison between our aCute software and the Triangle software is given in our website [52]. Here, we give a summary of the important differences and similarities between them.

The popular mesh generation software Triangle [106] is a robust implementation of the Delaunay refinement method [101]. The software used Ruppert's circumcenter insertion algorithm [99] in the first four releases (latest Triangle 1.4) and has been using Üngör's off-center insertion algorithm [118] in its latest two releases (Triangle versions 1.5 and 1.6). Here, we compare our software with Triangle version 1.4, mainly because it is a classical representation of Delaunay refinement algorithms and also off-centers are part of our unified Steiner point definition. However, comparing with version 1.6 would give similar results, except its advancing front type of algorithmic approach, and having slightly better small angle bounds (35°) and smaller output size (nearly half) than circumcenter-based algorithm.

- **Small angle quality:** aCute achieves premium quality in small angle by generating triangulations having smallest angle as high as 42°. On the other hand, this practical bound is only 34° for Triangle.
- **Simultaneous small and large angle quality:** Unlike Triangle, aCute can generate triangulations with non-trivial upper bounds on large angles, where the largest angles can be as low as 81°. Triangle only achieves trivial large angle bounds ($\pi - 2\alpha = 180^\circ - 2 * 34^\circ = 112^\circ$). This results in making aCute the first software generating such triangulations.
- **Algorithmic approach:** aCute follows an advancing front type of point insertion sequence, since it inserts locally optimal Steiner points described in Chapter 3 and 4 starting from the boundary. Conversely, Triangle follows circumcenter insertion, where no specific location is fixed in order. (See Figure 5-1.)
- **Output size:** Thanks to our efficient integration of refinement and relocation, aCute generates significantly smaller triangulations with the same quality than Triangle (usually half).
- **Speed:** Especially due to the improvement on the output size, aCute generates triangulations faster.

5.2 Future Work

Improving theoretical bounds. The theoretical termination and size complexity bounds given for the previous Delaunay refinement algorithms also apply for our algorithms (constraint angles up to $30°$), as we are more cautious in introducing short features. It would be interesting to prove the same theoretical (termination and size-optimality) bounds for $\alpha > 30°$. In addition, a careful analysis on our heuristic method generating no small no large angle triangulations might explain the good practical performance of our method. A guaranteed $[\alpha, \gamma]$ bound would be useful for future studies.

Alternative smoothing strategies. Alternative (more powerful but perhaps more expensive) mesh optimization algorithms can be integrated within our framework. The cost-benefit analysis of this replacement over our simple and efficient relocation strategy is also worth exploring as future research.

New boundary handling strategies. We adopted the same boundary handling strategy used in the standard Delaunay refinement algorithms [104]. An alternative boundary handling rule might be more effective.

Extension to other dimensions. Delaunay refinement is a popular technique for computing surface triangulations also [34]. We foresee that our algorithms can be easily extended for computing high quality and small size triangulations of two manifolds. Extension of the method to three dimensions is also a natural research direction.

Employing our smoothing strategy as post-refinement-smoothing. Our local smoothing strategy has been explored as a post-refinement-smoothing method in [55]. Preliminary results showed great potential for a powerful smoothing technique, especially for handling complex domains unlike other smoothing methods. Further developing our ideas for mesh improvement techniques could result in successful applications.

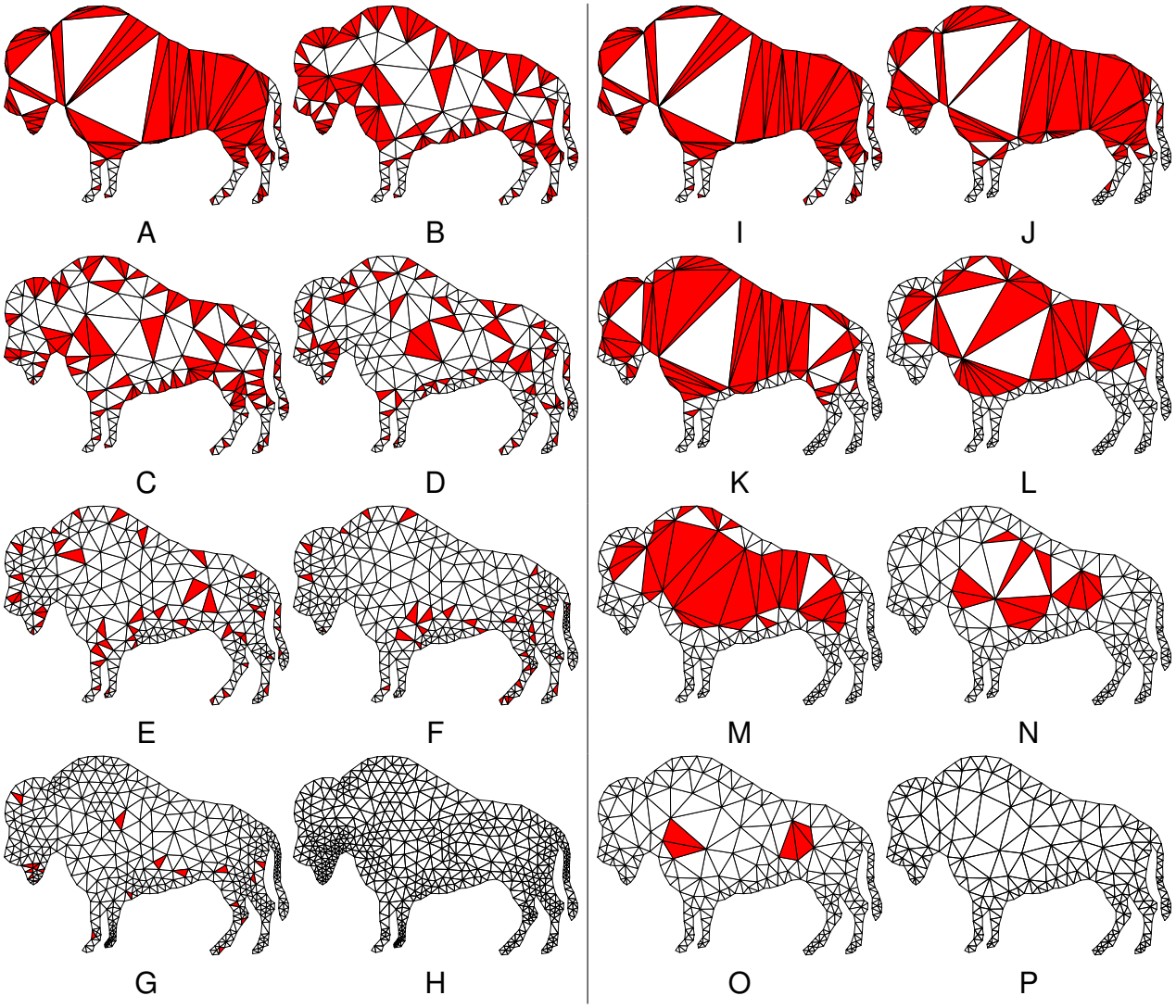


Figure 5-1. Algorithmic flow of Triangle and aCute for buffalo data set for $\alpha = 34°$. A - N) Triangle, 0, 30, 50, 100, 150, 200, 300, 412 Steiner points inserted, respectively. I - P) aCute, 0, 30, 50, 75, 85, 105, 110, 114 Steiner points inserted, respectively.

REFERENCES

[1] A. AGGARWAL, L. GUIBAS, J. SAXE, AND P. SHOR, *A linear-time algorithm for computing the Voronoi diagram of a convex polygon*, Discrete & Computational Geometry, 4 (1989), pp. 591–604.

[2] P. ALLIEZ, D. COHEN-STEINER, M. YVINEC, AND M. DESBRUN, *Variational tetrahedral meshing*, ACM Transactions on Graphics, 24 (2005), pp. 617–625.

[3] N. AMENTA, M. W. BERN, AND D. EPPSTEIN, *Optimal point placement for mesh smoothing*, Journal of Algorithms, 30 (1999), pp. 302–322.

[4] F. AURENHAMMER, *Voronoi diagrams - A survey of a fundamental geometric data structure*, ACM Computing Surveys, 23 (1991), pp. 345–405.

[5] F. AURENHAMMER AND Y. XU, *Optimal triangulations*, in Encyclopedia of Optimization 4, P.M.Pardalos and C.A.Floudas, eds., Kluwer Academic Publishing, 1999, pp. 160–166.

[6] I. BABUŠKA AND A. AZIZ, *On the angle condition in the finite element method*, SIAM Journal on Numerical Analysis, 13 (1976), pp. 214–227.

[7] B. S. BAKER, E. GROSSE, AND C. S. RAFFERTY, *Non-obtuse triangulation of polygons*, Discrete & Computational Geometry, 3 (1988), pp. 147–168.

[8] R. E. BANK AND R. K. SMITH, *Mesh smoothing using a posteriori error estimates*, SIAM Journal on Numerical Analysis, 34 (1997), pp. 979–997.

[9] M. BERN, D. DOBKIN, AND D. EPPSTEIN, *Triangulating polygons without large angles*, in Proceedings of the 8th Annual ACM Symposium on Computational Geometry, Berlin, Germany, 1992, pp. 222–231.

[10] M. BERN, H. EDELSBRUNNER, D. EPPSTEIN, S. A. MITCHELL, AND T. TAN, *Edge insertion for optimal triangulation*, Discrete & Computational Geometry, 10 (1993), pp. 47–65.

[11] M. BERN AND D. EPPSTEIN, *Mesh generation and optimal triangulation*, in Computing in Euclidean Geometry, F. K. Hwang and D.-Z. Du, eds., World Scientific, 1992, pp. 23–90.

[12] M. BERN AND D. EPPSTEIN, *Polynomial-size non-obtuse triangulation of polygons*, International Journal of Computational Geometry & Applications, 2 (1992), pp. 241–255.

[13] M. BERN, D. EPPSTEIN, AND J. GILBERT, *Provably good mesh generation*, Journal of Computer and System Sciences, 48 (1994), pp. 384–409.

[14] M. BERN, D. EPPSTEIN, AND S.-H. TENG, *Parallel construction of quadtrees and quality triangulations*, International Journal of Computational Geometry & Applications, 9 (1999), pp. 517–532.

[15] M. BERN, S. MITCHELL, AND J. RUPPERT, *Linear-size non-obtuse triangulation of polygons*, in Proceedings of the 10th Annual ACM Symposium on Computational Geometry, Stony Brook, NY, 1994, pp. 221–230.

[16] M. BERN AND P. PLASSMANN, *Mesh generation*, in Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, eds., Elsevier, 1998, pp. 291–332.

[17] H. BOROUCHAKI, L. PATRICK, AND P. GEORGE, *Parametric surface meshing using a combined advancing front generalized Delaunay approach*, International Journal for Numerical Methods in Engineering, 49 (2000), pp. 233–259.

[18] J. H. BRAMBLE AND M. ZLAMAL, *Triangular elements in finite element method*, Mathematics of Computation, 24 (1970), pp. 809–820.

[19] S. A. CANANN, J. R. TRISTANO, AND T. BLACKER, *Opti-smoothing: An optimization-driven approach to mesh smoothing*, Finite Elements in Analysis and Design, 13 (1993), pp. 185–190.

[20] S. A. CANANN, J. R. TRISTANO, AND M. L. STATEN, *An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes*, in 7th International Meshing Roundtable, Dearborn, MI, 1998, pp. 479–494.

[21] G. F. CAREY, *Computational Grids: Generation, adaptation and solution strategies*, Taylor & Francis, Washington, DC, 1997.

[22] C. CASSIDY AND G. LORD, *A square acutely triangulated*, Journal of Recreational Mathematics, 13 (1980), pp. 263–268.

[23] B. CHAZELLE, *Triangulating a simple polygon in linear time*, Discrete & Computational Geometry, 6 (1991), pp. 485–524.

[24] L. CHEN, *Mesh smoothing schemes based on optimal Delaunay triangulations*, in 13th International Meshing Roundtable, Williamsburg, VA, 2004, pp. 109–120.

[25] L. CHEN AND J. XU, *Optimal Delaunay triangulation*, Journal of Computational Mathematics, 22 (2004), pp. 299–308.

[26] S.-W. CHENG AND T. K. DEY, *Quality meshing with weighted Delaunay refinement*, in Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, 2002, pp. 137–146.

[27] S.-W. CHENG AND K.-H. LEE, *Quadtree decomposition, Steiner triangulation, and ray shooting*, in Proceedings of the 9th International Symposium on Algorithms and Computation, Taejon, Korea, 1998, pp. 367–376.

[28] L. CHEW, *Guaranteed-quality triangular meshes*, Tech. Rep. TR-89-983, Cornell University, 1989.

[29] L. CHEW, *Guaranteed-quality mesh generation for curved surfaces*, in Proceedings of the 9th Annual ACM Symposium on Computational Geometry, San Diego, CA, 1993, pp. 274–280.

[30] L. CHEW, N. CHRISOCHOIDES, AND F. SUKUP, *Parallel constrained Delaunay meshing*, in Trends in Unstructured Mesh Generation, S. A. Canann and S. Saigal, eds., ASME, 1997, pp. 89–96.

[31] L. P. CHEW, *Constrained Delaunay triangulations*, Algorithmica, 4 (1989), pp. 97–108.

[32] N. COLL, M. GUERRIERI, AND J. SELLARÉS, *Mesh modification under local domain changes*, in Proceedings of the 15th International Meshing Roundtable, Birmingham, AL, 2006, pp. 39–56.

[33] M. DE BERG, M. VAN KREVELD, M. OVERMARS, , AND O. SCHWARZKOPF, *Computational Geometry: Algorithms and Applications (2nd ed.)*, Springer-Verlag, New York, NY, 2000.

[34] T. DEY, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, Cambridge University Press, New York, NY, 2006.

[35] L. F. DIACHIN, P. M. KNUPP, AND T. S. M. ANS S. M. SHONTZ, *A comparison of two optimization methods for mesh quality improvement*, Engineering with Computers, 22 (2006), pp. 61–74.

[36] H. DINH, A. YEZZI, AND G. TURK, *Texture transfer during shape transformation*, ACM Transactions on Graphics, 24 (2005), pp. 289–310.

[37] Q. DU, V. FABER, AND M. GUNZBURGER, *Centroidal Voronoi tessellations: Applications and algorithms*, SIAM Review, 41 (1999), pp. 637–676.

[38] Q. DU AND M. GUNZBURGER, *Grid generation and optimization based on centroidal Voronoi tessellations*, Applied Mathematics and Computation, 133 (2002), pp. 591–607.

[39] Q. DU AND D. WANG, *Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations*, International Journal for Numerical Methods in Engineering, 56 (2002), pp. 1355–1373.

[40] Q. DU AND D. WANG, *Recent progress in robust and quality Delaunay mesh generation*, Journal of Computational and Applied Mathematics, 195 (2006), pp. 8–23.

[41] R. A. DWYER, *A faster divide and conquer algorithm for constructing Delaunay triangulation*, International Journal of Computer and Information Sciences, 2 (1987), pp. 137–151.

[42] M. ECK, T. DEROSE, T. DUCHAMP, H. HOPPE, M. LOUNSBERY, AND W. STUETZLE, *Multiresolution analysis of arbitrary meshes*, in Proceedings of the ACM SIGGRAPH '95, Los Angeles, CA, 1995, pp. 173–182.

[43] H. EDELSBRUNNER, *Triangulations and meshes in computational geometry*, Acta Numerica, 9 (2000), pp. 133–213.

[44] H. EDELSBRUNNER, *Geometry and topology for mesh generation*, Cambridge University Press, New York, NY, 2001.

[45] H. EDELSBRUNNER AND D. GUOY, *Sink insertion for mesh improvement*, in Proceedings of the 17th Annual ACM Symposium on Computational Geometry, Medford, MA, 2001, pp. 115–123.

[46] H. EDELSBRUNNER, X.-Y. LI, G. L. MILLER, A. STATHOPOULOS, D. TALMOR, S.-H. TENG, A. ÜNGÖR, AND N. WALKINGTON, *Smoothing cleans up slivers*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, Portland, OR, 2000, pp. 273–277.

[47] H. EDELSBRUNNER, T. S. TAN, AND R. WAUPOTITSCH, *An $O(n^2 log n)$ time algorithm for the min-max angle triangulation*, Journal on Scientific and Statistical Computing, 13 (1992), pp. 994–1008.

[48] D. EPPSTEIN, *Approximating the minimum weight triangulation*, Discrete & Computational Geometry, 11 (1994), pp. 163–191.

[49] D. EPPSTEIN, *Faster circle packing with applications to non-obtuse triangulation*, International Journal of Computational Geometry & Applications, 7 (1997), pp. 485–492.

[50] H. ERTEN AND A. ÜNGÖR, *Computing acute and non-obtuse triangulations*, in Proceedings of the 19th Canadian Conference on Computational Geometry, Ottawa, Canada, 2007, pp. 205–208.

[51] H. ERTEN AND A. ÜNGÖR, *Triangulations with locally optimal Steiner points*, in Proceedings of the Eurographics Symposium on Geometry Processing, Barcelona, Spain, 2007, pp. 143–152.

[52] H. ERTEN AND A. ÜNGÖR, *aCute: Premium Quality Meshing Software*, 2009. Retrieved September 27, 2009, from http://www.cise.ufl.edu/~ungor/aCute.

[53] H. ERTEN AND A. ÜNGÖR, *Quality triangulations with locally optimal Steiner points*, SIAM Journal of Scientific Computing, 31 (2009), pp. 2103–2130.

[54] H. ERTEN AND A. ÜNGÖR, *Triangulations without small and large angles*, in Proceedings of the 6th International Symposium on Voronoi Diagrams, Copenhagen, Denmark, 2009. To appear.

[55] H. ERTEN, A. ÜNGÖR, AND C. ZHAO, *Mesh smoothing algorithms for complex geometric domains*, in Proceedings of the 18th International Meshing Roundtable, Salt Lake City, UT, 2009, pp. 175–193.

[56] D. A. FIELD, *Laplacian smoothing and Delaunay triangulations*, Communications in Applied Numerical Methods, 4 (1988), pp. 709–712.

[57] S. FORTUNE, *A sweepline algorithm for Voronoi diagrams*, Algorithmica, 2 (1987), pp. 153–174.

[58] S. FORTUNE, *Voronoi diagrams and Delaunay triangulations*, in Computing in Euclidean Geometry, F. K. Hwang and D.-Z. Du, eds., World Scientific, 1992, pp. 225–265.

[59] L. A. FREITAG, *On combining Laplacian and optimization-based mesh smoothing techniques*, in Trends in Unstructured Mesh Generation, S. A. Canann and S. Saigal, eds., ASME, 1997, pp. 37–43.

[60] L. A. FREITAG, P. M. KNUPP, T. S. MUNSON, AND S. M. SHONTZ, *A comparison of optimization software for mesh shape-quality improvement problems*, in Proceedings of the 11th International Meshing Roundtable, 2002, pp. 29–40.

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

[62] M. R. GAREY, D. S. JOHNSON, F. P. PREPARATA, AND R. E. TARJAN, *Triangulating a simple polygon*, Information Processing Letters, 7 (1978), pp. 175–179.

[63] J. E. GOODMAN AND J. O'ROURKE, eds., *Handbook of Discrete and Computational Geometry*, CRC Press LLC, Boca Raton, FL, 1997.

[64] L. J. GUIBAS, D. E. KNUTH, AND M. SHARIR, *Randomized incremental construction of Delaunay and Voronoi diagrams*, Algorithmica, 7 (1992), pp. 381–413.

[65] L. J. GUIBAS AND J. STOLFI, *Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams.*, ACM Transactions on Graphics, 4 (1985), pp. 74–123.

[66] D. GUOY, *Tetrahedral mesh improvement, algorithms and experiments*, PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, 2001.

[67] T. HANGAN, J. ITOH, AND T. ZAMFIRESCU, *Acute triangulations*, Bulletin Math. de la Soc. des Sci. Math. de Roumanie, 43 (2000), pp. 279–286.

[68] S. HAR-PELED AND A. ÜNGÖR, *A time-optimal Delaunay refinement algorithm in two dimensions*, in Proceeding of the 21st Annual ACM Symposium on Computational Geometry, Pisa, Italy, 2005, pp. 228–236.

[69] B. HUDSON, G. MILLER, AND T. PHILLIPS, *Sparse Voronoi refinement*, in Proceedings of the 15th International Meshing Roundtable, Birmingham, AL, 2006, pp. 339–356.

[70] J.-I. ITOH, *Acute triangulations of sphere and icosahedron*, Josai Mathematical Monographs, 3 (2001), pp. 53–62.

[71] R. KIMMEL AND J. SETHIAN, *Computing geodesic paths on manifolds*, Proceedings of the National Academy of Sciences, 95 (1998), pp. 8431–8435.

[72] S. KOROTOV, M. KŘÍZEK, AND J. SOLC, *On a discrete maximum principle for linear FE solutions of elliptic problems with a non-diagonal coefficient matrix*, in Proceedings of the 4th International Conference on Numerical Analysis and Its Applications, Lozenetz, Bulgaria, 2008, pp. 384–391.

[73] C. LEE, *Automatic metric advancing front triangulation over curved surfaces*, Engineering Computations, 17 (2000), pp. 48–74.

[74] D. T. LEE AND B. J. SCHACHTER, *Two algorithms for constructing a Delaunay triangulation*, International Journal of Computer and Information Sciences, 9 (1980), pp. 219–242.

[75] X. LI, S. TENG, AND A. ÜNGÖR, *Biting: Advancing front meets sphere packing*, International Journal for Numerical Methods in Engineering, 49 (2000), pp. 61–81.

[76] X.-Y. LI AND S.-H. TENG, *Generating well-shaped Delaunay meshed in 3D*, in Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, MD, 2001, pp. 28–37.

[77] R. LOHNER, *Progress in grid generation via the advancing front technique*, Engineering with Computers, 12 (1996), pp. 186–199.

[78] R. LOHNER AND C. JUAN, *Generation of non-isotropic unstructured grids via directional enrichment*, Engineering with Computers, 49 (2000), pp. 219–232.

[79] H. MAEHARA, *Acute triangulations of polygons*, in Proceedings of the Japan Conference on Discrete & Computational Geometry, Tokyo, Japan, 2000, pp. 237–243.

[80] W. MANHEIMER, *Dissecting an obtuse triangle into acute triangles*, American Mathematical Monthly, 67 (1960), p. 923.

[81] D. MARCUM AND N. WEATHERILL, *Unstructured grid generation using iterative point insertion and local reconnection*, AIAA Journal, 33 (1995), pp. 1619–1625.

[82] E. A. MELISSARATOS AND D. L. SOUVAINE, *Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes*, in Proceedings of the 8th Annual ACM Symposium on Computational Geometry, Berlin, Germany, 1992, pp. 202–211.

[83] A. MIDDLEDITCH, T. STACEY, AND S. TOR, *Intersection algorithms for lines and circles*, ACM Transactions on Graphics, 8 (1989), pp. 25–40.

[84] G. MILLER, *A time efficient Delaunay refinement algorithm*, in Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, LA, 2004, pp. 400–409.

[85] G. L. MILLER, S. E. PAV, AND N. WALKINGTON, *When and why Delaunay refinement algorithms work*, International Journal of Computational Geometry & Applications, 15 (2005), pp. 25–54.

[86] G. L. MILLER, T. PHILLIPS, AND D. SHEEHY, *Size competitive meshing without large angles*, in Proceedings of the 34th International Colloquium on Automata, Languages and Programming, Wroclaw, Poland, 2007, pp. 655–666.

[87] G. L. MILLER, D. TALMOR, S.-H. TENG, AND N. WALKINGTON, *A Delaunay based numerical method for three dimensions: Generation, formulation, and partition*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, Las Vegas, NV, 1995, pp. 683–692.

[88] S. MITCHELL AND S. VAVASIS, *Quality mesh generation in higher dimensions*, SIAM Journal of Computing, 29 (2000), pp. 1334–1370.

[89] S. MITCHELL AND S. A. VAVASIS, *QMG: Mesh generation and related software*, 1995. Retrieved September 27, 2009, from http://www.cs.cornell.edu/home/vavasis/qmg-home.html.

[90] S. A. MITCHELL, *Refining a triangulation of a planar straight-line graph to eliminate large angles*, in Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1993, pp. 583–591.

[91] N. MUKHERJEE, *A hybrid, variational 3D smoother for orphaned shell meshes*, in Proceedings of the 11th International Meshing Roundtable, Ithaca, NY, 2002, pp. 379–390.

[92] V. N. PARTHASARATHY AND S. KODIYALAM, *A constrained optimization approach to finite element mesh smoothing*, Finite Elements in Analysis and Design, 9 (1991), pp. 309–320.

[93] E. PAV, *Delaunay Refinement Algorithms*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2003.

[94] S. PAV AND N. WALKINGTON, *Delaunay refinement with corner lopping*, in Proceedings of the 14th International Meshing Roundtable, San Diego, CA, 2005, pp. 165–181.

[95] P. P. PEBAY AND T. J. BAKER, *A comparison of triangle quality measures*, in Proceedings of the 10th International Meshing Roundtable, Newport Beach, CA, 2001, pp. 327–340.

[96] U. PINKALL AND K. POLTHIER, *Computing discrete minimal surfaces and their conjugates*, Experimental Mathematics, 2 (1993), pp. 15–36.

[97] F. P. PREPARATA AND M. I. SHAMOS, *Convex hulls of finite sets of points in two and three dimensions*, Communications of the ACM, 20 (1977), pp. 87–93.

[98] M. RIVARA AND N. HITSCHFELD, *LEPP-Delaunay algorithm: A robust tool for producing size-optimal quality triangulations*, in Proceedings of the 8th International Meshing Roundtable, South Lake Tahoe, CA, 1999, pp. 205–220.

[99] J. RUPPERT, *A new and simple algorithm for quality 2-dimensional mesh generation*, in Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms (SODA), Austin, TX, 1993, pp. 83–92.

[100] M. I. SHAMOS AND D. HOEY, *Closest point problems*, in Proceedings of the 16th IEEE Symposium on Foundations of Computer Science, Berkeley, CA, 1975, pp. 151–162.

[101] J. R. SHEWCHUK, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, in Proceedings of the 1st Workshop on Applied Computational Geometry, Philadelphia, PA, 1996, pp. 124–133.

[102] J. R. SHEWCHUK, *Delaunay refinement mesh generation*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

[103] J. R. SHEWCHUK, *Mesh generation for domains with small angles.*, in Proceedings of the 16th Annual ACM Symposium on Computational Geometry, New York, NY, 2000, pp. 1–10.

[104] J. R. SHEWCHUK, *Delaunay refinement algorithms for triangular mesh generation*, Computational Geometry: Theory and Applications, 22 (2002), pp. 21–74.

[105] J. R. SHEWCHUK, *What is a good linear element? - Interpolation, conditioning, and quality measures*, in Proceedings of the 11th International Meshing Roundtable, Ithaca, NY, 2002, pp. 115–126.

[106] J. R. SHEWCHUK, *Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator*, 2005. Retrieved September 27, 2009, from http://www.cs.cmu.edu/~quake/triangle.html.

[107] K. SHIMADA AND D. C. GOSSARD, *Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing*, in Proceedings of the 3rd ACM Symposium on Solid Modeling and Applications, Salt Lake City, UT, 1995, pp. 409–419.

[108] R. SIBSON, *Locally equiangular triangulations*, The Computer Journal, 21 (1978), pp. 243–245.

[109] B. SIMPSON, *How efficient are Delaunay refined meshes? An empirical study*, in Proceedings of the 15th International Meshing Roundtable, Birmingham, AL, 2006, pp. 215–237.

[110] D. A. SPIELMAN, S. H. TENG, AND A. ÜNGÖR, *Parallel Delaunay refinement: Algorithms and analyses*, in Proceedings of the 11th International Meshing Roundtable, Ithaca, NY, 2002, pp. 205–217.

[111] G. STRANG, *Approximation in the finite element method*, Numerical Mathematics, 19 (1972), pp. 81–98.

[112] G. STRANG AND G. J. FIX, *Approximation in the finite element method*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[113] P. SU AND R. L. DRYSDALE, *A comparison of sequential Delaunay triangulation algorithms*, in Proceedings of the 11th Annual ACM Symposium on Computational Geometry, Vancouver, B.C., Canada, 1995, pp. 61–70.

[114] T.-S. TAN, *An optimal bound for high-quality conforming triangulations*, Discrete & Computational Geometry, 15 (1996), pp. 169–193.

[115] R. E. TARJAN AND C. J. V. WYK, *An $O(nloglogn)$-time algorithm for triangulating a simple polygon*, SIAM Journal of Computing, 17 (1988), pp. 143–178.

[116] S.-H. TENG AND C. W. WONG, *Unstructured mesh generation: Theory, practice, and perspectives*, International Journal of Computational Geometry & Applications, 10 (2000), pp. 227–266.

[117] J. TOURNOIS, P. ALLIEZ, AND O. DEVILLERS, *Interleaving Delaunay refinement and optimization for 2D triangle mesh generation*, in Proceedings of the 16th International Meshing Roundtable, Seattle, WA, 2007, pp. 83–101.

[118] A. ÜNGÖR, *Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations*, in Proceedings of the Latin American Theoretical Informatics Symposium (LATIN), 2004, pp. 152–161.

[119] A. ÜNGÖR AND A. SHEFFER, *Pitching tents in space-time: Mesh generation for discontinuous Galerkin method*, Journal of Foundations of Computer Science, 13 (2002), pp. 201–221.

[120] E. VANDERZEE, A. N. HIRANI, D. GUOY, AND E. RAMOS, *Well-centered planar triangulation - an iterative approach*, in Proceedings of the 16th International Meshing Roundtable, Seattle, WA, 2007, pp. 121–138.

[121] T. S. VASSILEV, *Optimal Area Triangulation*, PhD thesis, University of Saskatchewan, Saskatoon, SK, Canada, 2007.

[122] D. WANG AND Q. DU, *Mesh optimization based on the centroidal Voronoi tessellation*, International Journal of Numerical Analysis & Modeling, 2 (2005), pp. 100–113.

[123] H. XU AND T. S. NEWMAN, *An angle-based optimization approach for 2D finite element mesh smoothing*, Finite Elements in Analysis and Design, 42 (2006), pp. 1150–1164.

[124] M. A. YERRY AND M. S. SHEPHARD, *A modified quadtree approach to finite element mesh generation*, IEEE Computer Graphics and Applications, 3 (1983), pp. 39–460.

[125] T. ZHOU AND K. SHIMADA, *An angle-based approach to two-dimensional mesh smoothing*, in Proceedings of the 9th International Meshing Roundtable, New Orleans, LA, 2000, pp. 373–384.

BIOGRAPHICAL SKETCH

Hale Erten received her Bachelor of Science degree in Computer Engineering from the Bogazici University, Istanbul Turkey in 2005. She pursued her Master of Science and Doctor of Philosophy degrees at the University of Florida from the department of Computer and Information Science and Engineering until her graduation in 2009. Her research interests include mesh generation, computational geometry, algorithms and graphics.