William & Mary
CHARTERED 1693

W&M ScholarWorks

Dissertations, Theses, and Masters Projects

Theses, Dissertations, & Master Projects

2007

Parallel generalized Delaunay mesh refinement

andrey Nikolayevich Chernikov
College of William & Mary - Arts & Sciences

Follow this and additional works at: https://scholarworks.wm.edu/etd

Part of the Computer Sciences Commons

Recommended Citation

Chernikov, andrey Nikolayevich, "Parallel generalized Delaunay mesh refinement" (2007). *Dissertations, Theses, and Masters Projects.* Paper 1539623328.
https://dx.doi.org/doi:10.21220/s2-4rx2-k272

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

Parallel generalized Delaunay mesh refinement

Andrey Nikolayevich Chernikov

Nalchik, Kabardino-Balkar Republic, Russia

Master of Science, Kabardino-Balkar State University, 2001
Bachelor of Science, Kabardino-Balkar State University, 1999

A Dissertation presented to the Graduate Faculty
of the College of William and Mary in Candidacy for the Degree of
Doctor of Philosophy

Department of Computer Science

The College of William and Mary
August 2007

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Andrey Chernikov

Andrey Nikolayevich Chernikov

Approved by the Committee, June 2007

Committee Chair
Associate Professor Nikos Chrisochoides
Computer Science, The College of WIlliam and Mary

Robert E Noonan

Professor Robert Noonan
Computer Science, The College of WIlliam and Mary

Weizhen Mao

Associate Professor Weizhen Mao
Computer Science, The College of WIlliam and Mary

Professor Gary Miller
Computer Science, Carnegie Mellon University

Associate Professor Dimitrios Nikolopoulos
Computer Science, Virginia Tech

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ABSTRACT

The modeling of physical phenomena in computational fracture mechanics, computational fluid dynamics and other fields is based on solving systems of partial differential equations (PDEs). When PDEs are defined over geometrically complex domains, they often do not admit closed form solutions. In such cases, they are solved approximately using discretizations of domains into simple elements like triangles and quadrilaterals in two dimensions (2D), and tetrahedra and hexahedra in three dimensions (3D). These discretizations are called finite element meshes. Many applications, for example, real-time computer assisted surgery, or crack propagation from fracture mechanics, impose time and/or mesh size constraints that cannot be met on a single sequential machine. As a result, the development of parallel mesh generation algorithms is required.

In this dissertation, we describe a complete solution for both sequential and parallel construction of guaranteed quality Delaunay meshes for 2D and 3D geometries. First, we generalize the existing 2D and 3D Delaunay refinement algorithms along with theoretical proofs of mesh quality in terms of element shape and mesh gradation. Existing algorithms are constrained by just one or two specific positions for the insertion of a Steiner point inside a circumscribed disk of a poorly shaped element. We derive an entire 2D or 3D region for the selection of a Steiner point (i.e., infinitely many choices) inside the circumscribed disk. Second, we develop a novel theory which extends both the 2D and the 3D Generalized Delaunay Refinement methods for the concurrent and mathematically guaranteed independent insertion of Steiner points. Previous parallel algorithms are either reactive relying on implementation heuristics to resolve dependencies in parallel mesh generation computations or require the solution of a very difficult geometric optimization problem (the domain decomposition problem) which is still open for general 3D geometries. Our theory solves both of these drawbacks. Third, using our generalization of both the sequential and the parallel algorithms we implemented prototypes of practical and efficient parallel generalized guaranteed quality Delaunay refinement codes for both 2D and 3D geometries using existing state-of-the-art sequential codes for traditional Delaunay refinement methods. On a heterogeneous cluster of more than 100 processors our implementation can generate a uniform mesh with about a billion elements in less than 5 minutes. Even on a workstation with a few cores, we achieve a significant performance improvement over the corresponding state-of-the-art sequential 3D code, for graded meshes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents

Abstract iii

Acknowledgments vi

List of Tables vii

List of Figures ix

1 Introduction 2
- 1.1 Motivation and Prior Work . . . 2
- 1.2 Contributions of This Dissertation . . . 12

2 Generalized Delaunay Refinement Using Selection Disks 14
- 2.1 Delaunay Refinement Background . . . 14
- 2.2 Two-Dimensional Generalized Delaunay Refinement . . . 21
  - 2.2.1 Proof of Termination with Selection Disks of Type I . . . 21
  - 2.2.2 An Example of a Point Selection Strategy . . . 28
  - 2.2.3 Proof of Good Grading and Size Optimality with Selection Disks of Type II . . . 33
- 2.3 Three-Dimensional Generalized Delaunay Refinement . . . 40
  - 2.3.1 Proof of Termination with Selection Disks of Type I . . . 40
  - 2.3.2 An Example of a Point Selection Strategy . . . 44
  - 2.3.3 Proof of Good Grading with Selection Disks of Type II . . . 47

3 Parallel Generalized Delaunay Refinement 53
- 3.1 Delaunay-Independence Criterion . . . 54
- 3.2 Two-Dimensional Parallel Uniform Delaunay Refinement . . . 57
  - 3.2.1 Sufficient Condition of Delaunay-Independence . . . 57
  - 3.2.2 Circumradius Upper Bound Loop Invariant . . . 59
  - 3.2.3 Adjusting the Degree of Concurrency . . . 63
  - 3.2.4 Coarse Grain Partitioning . . . 66
  - 3.2.5 Implementation and Evaluation . . . 71
- 3.3 Two-Dimensional Parallel Graded Delaunay Refinement . . . 77
  - 3.3.1 Sufficient Condition of Delaunay-Independence . . . 77
  - 3.3.2 Quadtree Construction . . . 81
  - 3.3.3 Implementation and Evaluation . . . 86

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4 Three-Dimensional Parallel Graded Delaunay Refinement 96
    3.4.1 Sufficient Condition of Delaunay-Independence 96
    3.4.2 Octree Construction 104
    3.4.3 Tetrahedron Quality Measures 111
    3.4.4 Implementation and Evaluation 114

4 Conclusions **119**
4.1 Summary and Extensions 119
4.2 Publications 123

Bibliography **125**

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACKNOWLEDGMENTS

First and foremost, I am grateful to my advisor Professor Nikos Chrisochoides for all the ways he supported me for the last six years. His wisdom, patience, and encouragement have made the writing of this dissertation possible. I thank the dissertation committee members Professors Gary Miller, Dimitrios Nikolopoulos, Weizhen Mao, and Robert Noonan for their helpful feedback and directions. I thank Professor George Karniadakis for the bat model, Professor Suchuan Dong for the flow model, Dr. Hang Si for the Tetgen software, and Professor Jonathan Shewchuk for the numerous papers, talks, and software. I thank all my research collaborators for helpful discussions, and in particular Andriy Kot for the shared-memory port of the uniform parallel Delaunay refinement algorithm. I thank the professors of the College of William and Mary whose courses helped me learn a great deal of what I do know and made me aware of much more that I do not know. I gratefully acknowledge all the hard work that the professors of the Kabardino-Balkar State University put into teaching me the foundations of science. I thank my Masters and Bachelors theses advisors, Professors Petr Zilberman and Leila Shautsukova, for introducing me to research and teaching me to appreciate the value of scholarly publication. I thank my conversation partner and good friend William and Mary Professor John Noell Moore for his continuous (and oftentimes iterative) effort in teaching me the proper English language, as well as helping me in the development of a wider perspective on innumerable aspects of American culture. I thank Sensei Steve Pohle of Newport News for his insightful karate classes which not only helped me clear my mind after the long research hours, but also introduced me to a different learning dimension. I also thank the former residents of the Graduate Complex (graduate student dormitory) for their cheerful company and friendship, in particular Andriy Fedorov, Andriy Kot, Alexei Matveev, and Lena Mingazova. I thank the Computer Science Department's administrative office and the Global Education office of the College for their professionalism in helping me navigate through mazes of paperwork. This dissertation work was performed using computational facilities at the College of William and Mary which were enabled by grants from Sun Microsystems, the National Science Foundation, and Virginia's Commonwealth Technology Research Fund. This work was supported by the following grants: EIA-9972853, EIA-0203974, ACI-0312980, and CNS-0521381.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

2.1 Mesh quality comparison for three two-dimensional point insertion strategies, no area bound is used . . . 30

2.2 The comparison of the number of triangles generated by three two-dimensional point insertion strategies, with an area bound . . . 32

2.3 The number of tetrahedra produced with the use of different point selection methods for three models in three dimensions . . . 46

3.1 Uniform parallel Delaunay refinement of a mesh of a unit square, time measurements for the scaled and the fixed workload . . . 73

3.2 A breakdown of the execution time for a single processor in the case of the unit square . . . 74

3.3 Uniform parallel Delaunay refinement of a mesh of the pipe model, time measurements for the scaled and the fixed workload . . . 74

3.4 Uniform parallel Delaunay refinement of a mesh of the pipe model, time measurements for the distributed and the shared memory implementations 74

3.5 The total time spent by the three-dimensional PGDR code on refining the meshes of the unit cube and the bat models . . . 118

4.1 A qualitative comparison of practical parallel Delaunay mesh generation methods. . . . 120

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

1.1 The use of the selection disk for the construction of boundary conformal meshes . . . 5

2.1 Encroachment in three dimensions . . . 17
2.2 Delaunay refinement with the off-centers and the Type I selection disk for the insertion of a Steiner point . . . 22
2.3 Flow diagram illustrating the relationship between the insertion radius of a vertex and its parent in two dimensions . . . 26
2.4 A Steiner point within a selection disk of a poor quality triangle and an optimization-based method for the selection of a Steiner point . . . 28
2.5 An upper part of a model of cylinder flow and a pipe cross-section model 30
2.6 The parts of the pipe mesh generated with different point insertion strategies 31
2.7 Selection of a Steiner point by a δ-graded Delaunay refinement algorithm 33
2.8 An illustration of the relationship between the insertion radii of Steiner points, in the case of encroachment in three dimensions . . . 42
2.9 Flow diagram illustrating the relationship between the insertion radius of a vertex and that of its parent in three dimensions . . . 44
2.10 A tetrahedron with high circumradius-to-shortest edge ratio . . . 45

3.1 Challenges in parallel Delaunay refinement . . . 54
3.2 Intersecting and adjacent cavities of Steiner points . . . 57
3.3 The circumradius of a new triangle cannot be larger than the circumradii of the existing triangles . . . 60
3.4 The possible relations among the circumradii of the triangles during Delaunay refinement . . . 61
3.5 A sequential Delaunay refinement algorithm with the restriction on the region for the insertion of Steiner points . . . 67
3.6 The uniform parallel Delaunay meshing algorithm . . . 68
3.7 A regenerative cooled pipe geometry . . . 72
3.8 Snapshots of the pipe mesh distributed among four processors . . . 75
3.9 A schedule of the uniform parallel Delaunay refinement and communication on distributed memory . . . 76
3.10 An edge shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, first case . . . 78
3.11 An edge shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, second case . . 79

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.12 An example of a buffer zone 83
3.13 Splitting constrained segments and strong Delaunay-independence 83
3.14 All possible positions, up to symmetry, of two Steiner points relative to the buffer zone of one of the points 84
3.15 The Parallel Generalized Delaunay Refinement algorithm executed by the master thread 87
3.16 The Parallel Generalized Delaunay Refinement algorithm executed by each of the refinement threads 88
3.17 The circumdisk of a new triangle is a subset of the union of the circumdisks of the existing triangles 90
3.18 The meshes of the pipe cross-section model generated with PGDR and with `Triangle` 92
3.19 The meshes of the key model generated with PGDR and with `Triangle`. . 93
3.20 The cylinder flow model and its mesh generated with PGDR 94
3.21 The total running time of PGDR code spent on refining the pipe model, for different granularity values 95
3.22 A three dimensional disk and its reflection cut by a plane 97
3.23 A face at the boundary of the cavity of a Steiner point, first case 99
3.24 A face at the boundary of the cavity of a Steiner point, second case 100
3.25 A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, first case . . . 101
3.26 A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, second case . . 102
3.27 The buffer zone of an octree leaf 105
3.28 Delaunay-separateness in three dimensions 106
3.29 Leaf Delaunay-separateness 107
3.30 A sliver tetrahedron 112
3.31 The diagram of the design of our parallel Delaunay refinement software . 115
3.32 A wireframe model of a flying bat 116
3.33 The bat inside a box 117
3.34 A nonuniform mesh of the bat inside a box 118

4.1 Examples of the approaches for choosing Steiner points within selection disks of skinny triangles. 121

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

PARALLEL GENERALIZED DELAUNAY MESH REFINEMENT

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

1.1 Motivation and Prior Work

Delaunay refinement is a popular technique for generating triangular and tetrahedral meshes for use in the finite element method and interpolation in various numeric computing areas. Among the reasons of its popularity is the amenability of the method to rigorous mathematical analysis, which allows to derive guarantees on the quality of the elements in terms of circumradius-to-shortest edge ratio, the gradation of the mesh, and the termination of the algorithm.

The field of sequential guaranteed quality Delaunay refinement has been extensively studied, see [24, 32, 35, 39, 62, 70, 77] and the references therein. However, new ideas and improvements keep being introduced. One of the basic questions is where to insert additional (so-called Steiner) points into an existing mesh in order to improve the quality of the elements. Frey's [37], Ruppert's [70], and early Chew's [24] algorithms use circumcenters of poor quality triangles. Later, Chew [25] suggested to use randomized insertion of *near-circumcenter* points for three-dimensional Delaunay refinement, with

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the goal of avoiding slivers.

Li and Teng [56, 57] extended the work in [25] by defining a picking sphere with a radius which is a constant multiple of the circumradius of the element. They use two different rules for eliminating the elements with large radius-edge ratio and for eliminating the slivers. In particular, in [56] the rules are defined as follows: "Add the circumcenter c_τ of any d-simplex with a large $\rho(\tau)$" and "For a sliver-simplex τ, add a good point $p \in \mathcal{P}(\tau)$", where $\rho(\tau)$ is the radius-edge ratio, $\mathcal{P}(\tau)$ is the picking region of simplex τ, and the good point is found by a constant number of random probes. The authors in [56] prove that their algorithm terminates and produces a well graded mesh with good radius-edge ratio and without slivers. In this dissertation, we define Type II selection disks similarly to the picking region in [56]. We extend the proofs in [56] to show that any point (not only the circumcenter) from the selection disk (picking region) can be used to eliminate the elements with large radius-edge ratios. We do not address the problem of sliver elimination, however, our work can be used in conjunction with the sliver removal procedure from [56] such that the Delaunay refinement algorithm can choose any points from the selection disks (picking regions) throughout both the stage of the construction of a good radius-edge ratio mesh ("almost good mesh" [56]) and the stage of sliver removal.

Recently, Üngör [86, 87] proposed to insert specially chosen points which he calls *off-centers*; this method allows to produce smaller meshes in practice and it was implemented in the popular sequential mesh generation software `Triangle` [73]. We expect that other optimization techniques can be used to select positions for new points. Indeed, in Subsections 2.2.2 and 2.3.2 we give examples of point placement strategies

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

which in some cases allow to achieve even smaller meshes than the off-center method, albeit at significant computation cost. Since one would not like to redesign the parallel algorithm and software to accommodate each of the point placement techniques, here we generalize the sequential Delaunay refinement approaches and develop a framework which allows to use custom point selection strategies. In particular, we derive two types of *selection disks* for the position of a new point with respect to a poor quality triangle or tetrahedron. We prove that any point placement technique with the only restriction that it chooses Steiner points inside selection disks of Type I will terminate and, furthermore, if it chooses Steiner points inside selection disks of Type II, will produce a well-graded mesh. While the use of Chew's [25] *picking-sphere* is restricted to produce only meshes with constant density, the use of our Type II selection disk allows to obtain graded meshes which are also size-optimal in two dimensions.

One of the important applications of the flexibility offered by the use of selection disks is in conforming the mesh to the boundary between different materials. The advantages are especially pronounced in medical imaging, when the boundaries between different tissues are blurred, see Figure 1.1(left). In this case, after the image is segmented, instead of a clear separation, we have a boundary zone of some none-negligible width, see Figure 1.1(right). Then the goal of the mesh generation step is to avoid creating edges that would intersect the boundary, which can be achieved by inserting Steiner points inside the boundary zone.

Many applications, like real-time computer assisted surgery and crack propagation from fracture mechanics, impose time and/or mesh size constraints that cannot be met on a single sequential machine. As a result, the development of parallel mesh gener-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

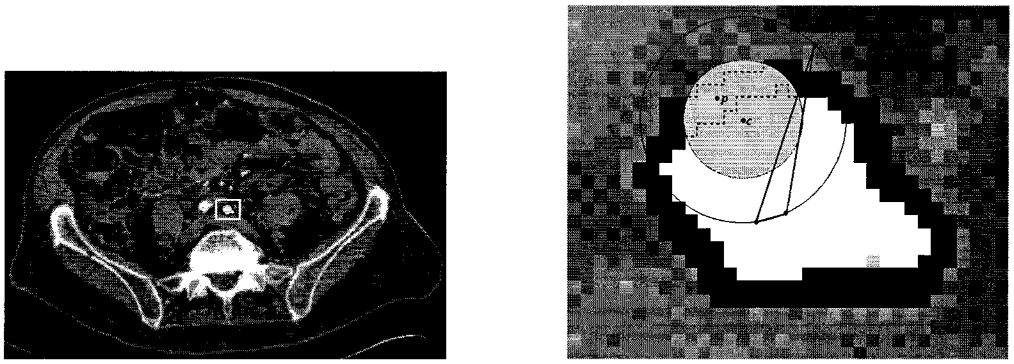


Figure 1.1: The use of the selection disk for the construction of boundary conformal meshes. **(Left)** An MRI scan showing a cross-section of a body. **(Right)** A zoom-in of the selected area containing an artery: the inside is white, the outside has different shades of gray and the black zone is an approximate boundary between these regions. The standard Delaunay refinement algorithm would insert the circumcenter c. However, in order to construct a mesh which conforms to the boundary, another point (p) would be a better choice.

ation algorithms is required. There are four important requirements that are usually imposed on parallel mesh generation algorithms: *stability*, *scalability*, *efficient domain decomposition* and *code re-use*. *Stability* refers to the fact that distributed meshes retain the same quality of the elements and partition properties as the sequentially generated and partitioned meshes [27]. *Scalability* is understood as the ability of the algorithm and software to achieve speedup proportional to the number of processors [54]. The *efficiency of domain decomposition* can be measured in terms of its computation costs and the appropriateness of the resulting subdomains for the selected mesh generation algorithm. Finally, *code reuse* allows to leverage the ever evolving basic sequential meshing libraries which are now mature and highly optimized.

The parallel mesh generation procedures can be classified based on the degree of coupling among the subproblems, see [28] for a survey on parallel mesh generation. This coupling determines the intensity of the communication and the degree of dependency (or synchronization). This dissertation completes a series of four different parallel mesh

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

generation classes of methods that were developed in our group and in collaboration:

1. tightly coupled published in [27,66],
2. partially coupled presented here and also published in [16–21],
3. weakly coupled published in [22,26], and
4. decoupled in [12,43,44,58–60].

Nave, Chrisochoides, and Chew [66] presented the first practical provably-good parallel mesh refinement algorithm for polyhedral domains, which is tightly coupled. They addressed the stability, to some success the scalability, but not the code re-use requirement. The algorithm in [66] guarantees stability by simultaneously partitioning and refining the interface surfaces and interiors of the subdomains at the cost of high communication and code re-structuring of the Delaunay refinement kernel: refinement due to a point insertion can propagate across processor boundaries. Chrisochoides and Nave in [27] restructured the sequential Bowyer-Watson kernel [8,90] using the speculative execution model [45] in order to tolerate high communication latencies. The speculative execution model allows setbacks to occur if the changes caused by the concurrent insertion of two or more points are incompatible. Although more than 80% of the overhead due to remote data gather operations is masked, experimental data in [27] suggest $\mathcal{O}(\log P)$ speedup, P being the number of processors. In this dissertation, we provide sufficient conditions, which guarantee the independence of simultaneously inserted points, and as a result, allow to eliminate setbacks and to avoid intensive communication. Moreover, our results are not based on expensive coloring techniques which require to construct and maintain a conflict graph (cavity graph): when a new point is inserted,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

all elements that have this point within their circumscribed disk are eliminated which creates a *cavity*. In the cavity graph, each cavity is represented by a vertex and two conflicting cavities are represented by an edge.

The elimination of setbacks renders the restructuring of the sequential codes unnecessary, which increases the possibilities to reuse the existing sequential mesh generation and refinement libraries. Code reuse is very important; it took three years to develop a parallel guaranteed quality Delaunay mesh generation code for the algorithm presented in [66] which is based on Delaunay algorithms developed five to six years before. This means that the parallel mesher at the time it is delivered is based on nine to ten year old technology. Moreover, improvements in terms of quality, speed, and functionality are open ended and permanent.

In order to eliminate communication and synchronization and maximize code reuse, Linardakis and Chrisochoides [58, 59] presented a Parallel Domain Decoupling Delaunay (PD^3) method for two-dimensional domains. The PD^3 is based on the idea of decoupling the individual submeshes so that they can be meshed independently with zero communication and synchronization. In the past similar parallelizations of Delaunay triangulation algorithms and implementations of Delaunay based mesh generators were presented in [6, 7, 38]. However, in [58, 59] the authors solve some of the drawbacks and improve upon the previously published methods. After the decomposition of the domain, the PD^3 constructs a special zone around the interfaces of the submeshes. The authors prove that Delaunay meshers will not insert any new points within a zone around the subdomain interfaces, i.e., the sequential Delaunay meshing of the individual submeshes can terminate without inserting any new points on the interfaces and thus eliminate com-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

munication and modifications of the sequential codes. This way, the problem of parallel meshing is reduced to a proper domain decomposition and a discretization of interfaces. The domain decomposition problem for parallel mesh generation is formulated as follows [28, 59, 60]. Given a domain $\Omega \subset \mathbb{R}^n$, construct the separators $S_{ij} \subset \mathbb{R}^{n-1}$, such that the domain is decomposed into *subdomains* Ω_i:

$$\Omega = \bigcup_{i=1}^{N} \Omega_i, \quad \partial\Omega_i \cap \partial\Omega_j = S_{ij}, \quad i, j = 1, \dots, N, \quad i \neq j,$$

where $\partial\Omega_i$ is the boundary of subdomain i, while the separators do not create very small angles and other features that will force the degradation of mesh quality. Of course, one has to show that the domain is not over-refined because of a predefined discretization of the interfaces. Experimental data from [58, 59] show very small over-refinement. The construction of the Medial Axis in [58, 59] is based on approximating it from an initial constrained Delaunay triangulation of the domain; an alternative way is to derive its exact continuous representation from the boundary representation only [13]. However, the construction of the Medial Axis for three-dimensional geometries is a much more challenging and still open problem [30, 72].

An idea of updating partition boundaries when the inserted points happen to be close to them was initially presented by Chew, Chrisochoides, and Sukup [26] as a Parallel Constrained Delaunay Meshing (PCDM) algorithm. In PCDM, the edges on the boundaries of submeshes are fixed (constrained), and if a new point encroaches upon a constrained edge, another point is inserted in the middle of this edge instead. As a result, a split message is sent to the neighboring processor, notifying that it also has to insert the midpoint of the shared edge. This approach requires the construction of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

separators that will not compromise the quality of the final mesh, and allows only for limited code reuse, since one has to handle split messages and partial cavity expansions within the Bowyer-Watson kernel. In [22] we presented algorithm improvements for the PCDM, e.g., the elimination of global synchronization, message aggregation, the use of the termination detection algorithm described by Dijkstra [33], and an efficient representation of split points, as well as a new implementation that utilizes a number of newly available software, such as the Medial Axis Domain Decomposition (MADD) library [60], the Metis graph partitioning library [50], the `Triangle` Delaunay triangulator and mesh generator [73], and the Robust Predicates [74]. Moreover, we evaluated the algorithm and its most recent implementation on a large cluster of workstations with more than 100 nodes. As a result, more than one billion triangles can be constructed in less than three minutes. However, the PCDM algorithm relies on the domain decompositions produced by the MADD software, and its extension to three dimensions is subject to the availability of a three-dimensional domain decomposer.

In addition to parallel mesh generation methods, there is a class of parallel triangulation methods. While the mesh generation problem deals with the selection of points which are not given in the input to achieve required mesh quality, the problem of triangulation is to construct a mesh for a pre-defined point set. A streaming approach to the triangulation problem was implemented by Isenburg, Liu, Shewchuk, and Snoeyink [42]. They achieve large performance gains by using a spatial finalization technique and manage to compute a billion triangle mesh from 500 million points of LIDAR data on a laptop in 48 minutes. A divide-and-conquer projection-based parallel Delaunay triangulation algorithm was developed by Blelloch, Hardwick, Miller, and Talmor [6, 7]. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

work by Kadow and Walkington [46–48] extended the work of Blelloch et al. for parallel mesh generation and further eliminated the sequential step for constructing an initial mesh, however, all potential conflicts among concurrently inserted points are resolved sequentially through global synchronization which in [47] is implemented by running a dedicated processor.

De Cougny, Shephard, and Ozturan [31] use an underlying octree to aid in parallel three-dimensional mesh generation. After the generation of the octree and template meshing of the interior octants, their algorithm connects a given surface triangulation to the interior octants using face removal. The face removal procedure eliminates problems due to the small distance between the interior quadrants and boundary faces, by defining "an entity too close to the boundary triangulation" and "using the distance of about one-half the octant edge length as the minimum" [31]. We explore a somewhat similar question in the context of Delaunay refinement and derive precise distances that are necessary between the interiors and the boundaries of concurrently refined subdomains.

Löhner and Cebral [61] developed a parallel advancing front scheme. They use an octree to delimit the zones where elements can be introduced concurrently and set the edge length of the smallest octree box to be of the order of 20 to 50 times the specified element size. They implement a "shift and regrid" technique with the shift distance determined by $\min(0.5s_{min}, 2.0d_{min})$, where s_{min} is the minimum box size in which elements are to be generated, and d_{min} is the minimum element size in the active front. These distances are likely to work well in the case of advancing front meshing, when there is a clear distinction between triangulated and empty areas, however, Delaunay refinement, in addition to maintaining a mesh which at all times covers the entire domain,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

also requires that all triangle circumcenters be empty of mesh points.

Edelsbrunner and Guoy [36] studied the possibility of parallel insertion of independent points. They define the prestar of point x as the difference between the closure of the set of tetrahedra whose circumspheres enclose x and the closure of the set of remaining tetrahedra. Further, they define points x and y as independent if the closures of their prestars are disjoint. In Section 3.1 we prove a similar condition of point independence. The difference between the independence condition in [36] and our initial criterion is that our formulation is less restrictive: under certain conditions it allows the prestars (we use the word cavity) to share a point. However, computing the prestars (cavities) and their intersections for all candidate points is very expensive. That is why we do not use coloring methods that are based on the cavity graphs and we prove sufficient conditions, which allow to use only the regions that the points belong to, to check whether they are independent. The minimum separation distance argument in [36] is used to derive the upper bound on the number of inserted vertices and prove termination, but does not ensure point independence. In addition, we propose a simple data (block) decomposition scheme for scheduling parallel point insertion for both distributed and shared memory implementations. In [36], a shared memory algorithm based on finding the maximal independent set of points is used.

Finally, Spielman, Teng, and Üngör [81] presented the first theoretical analysis of the complexity of parallel Delaunay refinement algorithms. However, the assumption is that the global mesh is completely retriangulated each time a set of independent points is inserted [83]. In [82] the authors extended their analysis.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.2 Contributions of This Dissertation

1. Generalization of the existing two-dimensional (2D) and three-dimensional (3D) Delaunay refinement algorithms along with theoretical proofs of mesh quality in terms of element shape and mesh gradation. Existing algorithms are constrained by just one or two specific positions for the insertion of a Steiner point inside a circumscribed disk of a poorly shaped element. We derive an entire 2D or 3D region for the selection of a Steiner point (i.e., infinitely many choices) inside the circumscribed disk.

2. The development of a novel theory which extends both the 2D and the 3D Generalized Delaunay Refinement methods (see 1) for the concurrent and mathematically guaranteed independent insertion of Steiner points. Previous parallel algorithms are either reactive relying on implementation heuristics to resolve dependencies in parallel mesh generation computations or require the solution of a very difficult geometric optimization problem (the domain decomposition problem) which is still open for general 3D geometries. Our theory solves both of these drawbacks.

3. Using our generalization of both the sequential and the parallel algorithms (see 1 and 2) we implemented prototypes of practical and efficient parallel generalized guaranteed quality Delaunay refinement codes for both 2D and 3D geometries using existing state-of-the-art sequential codes for traditional Delaunay refinement methods. On a heterogeneous cluster of more than 100 processors our implementation can generate a uniform mesh with about a billion elements in less than 5 minutes. Even on a workstation with a few cores, we achieve a significant perfor-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

mance improvement over the corresponding state-of-the-art sequential 3D code, for graded meshes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

Generalized Delaunay Refinement Using Selection Disks

2.1 Delaunay Refinement Background

Throughout this dissertation we will assume that the input domain Ω is described by a Planar Straight Line Graph (PSLG) in two dimensions, or a Planar Linear Complex (PLC) in three dimensions [70, 75–77]. A PSLG (PLC) $\mathcal{X}$ consists of a set of vertices, a set of straight line segments, and (in three dimensions) a set of planar facets. Each element of $\mathcal{X}$ is considered *constrained* and must be preserved during the construction of the mesh, although it can be subdivided into smaller elements. The vertices of the $\mathcal{X}$ must be a subset of the final set of vertices in the mesh.

Let the mesh $\mathcal{M}_{\mathcal{X}}$ for the given PSLG (PLC) $\mathcal{X}$ consist of a set $V = \{p_i\}$ of vertices and a set $T = \{t_i\}$ of elements which connect vertices from V. The elements are either triangles in two dimensions or tetrahedra in three dimensions. We will denote the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

triangle with vertices p_u, p_v, and p_w as $\triangle(p_u p_v p_w)$ and the tetrahedron with vertices p_k, p_l, p_m, and p_n as $\tau(p_k p_l p_m p_n)$. We will use the symbol $e(p_i p_j)$ to represent the edge of the mesh which connects points p_i and p_j, and the symbol $\mathcal{L}(p_i p_j)$ to represent the straight line segment connecting points which are not necessarily part of the mesh.

There are two commonly used parameters that control the quality of mesh elements: an upper bound $\bar{\rho}$ on the circumradius-to-shortest edge ratio, which in two dimensions is equivalent to a lower bound $\bar{A}$ on a minimal angle [63, 75]:

$$\bar{\rho} = \frac{1}{2\sin\bar{A}}, \tag{2.1}$$

and an upper bound on the element area (or volume) $\bar{\mathcal{A}}$. We will denote the circumradius-to-shortest edge ratio of element t as $\rho(t)$ and the area (or volume) of t as $\mathcal{A}(t)$. The former upper bound is usually fixed and given by a constant value, while the latter can vary and be controlled by some user-defined grading function $\bar{\mathcal{A}}(\cdot)$, which can be defined either over the set of mesh elements or over Ω, depending on the implementation. As a special case, the grading function can also be constant.

Let us call the open disk corresponding to a triangle's circumscribed circle or to a tetrahedron's circumscribed sphere its *circumdisk*. We will use symbols $\bigcirc(t)$ and $r(t)$ to represent the circumdisk and the circumradius of t, respectively. A mesh is said to satisfy the *Delaunay property* if the circumdisk of every element does not contain any of the mesh vertices [32].

Traditional Delaunay mesh generation algorithms start with the construction of the initial mesh, which conforms to $\mathcal{X}$, and then refine this mesh until the element quality constraints are met. In this dissertation, we focus on parallelizing the Delaunay refine-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ment stage, which is the most memory- and computation-expensive stage for parallel Delaunay mesh generation [17]. The general idea of Delaunay refinement is to insert additional (Steiner) points inside the circumdisks of poor quality elements, which causes these elements to be destroyed, until they are gradually eliminated and replaced by better quality elements.

We will extensively use the notion of *cavity* [39] which is the set of elements in the mesh whose circumdisks include a given point p. We will denote $\mathcal{C}_{\mathcal{M}}(p)$ to be the cavity of p with respect to mesh $\mathcal{M}$ and $\partial\mathcal{C}_{\mathcal{M}}(p)$ to be the set of boundary edges in two dimensions (or triangles in three dimensions) of the cavity, i.e., the edges or the triangles which belong to only one element in $\mathcal{C}_{\mathcal{M}}(p)$. When $\mathcal{M}$ is clear from the context, we will omit the subscript. For our analysis, we will use the Bowyer-Watson (B-W) point insertion algorithm [8, 90], which can be written shortly as follows:

$$\begin{aligned} V^{n+1} &\leftarrow V^n \cup \{p\}, \\ T^{n+1} &\leftarrow T^n \setminus \mathcal{C}_{\mathcal{M}^n}(p) \cup \{(p\xi) \mid \xi \in \partial\mathcal{C}_{\mathcal{M}^n}(p)\}, \end{aligned} \tag{2.2}$$

where ξ is an edge in two dimensions or a triangle in three dimensions, while $\mathcal{M}^{n+1} = (V^{n+1}, T^{n+1})$ and $\mathcal{M}^n = (V^n, T^n)$ represent the mesh before and after the insertion of p, respectively. The set of newly created elements forms a *ball* [39] of point p, denoted $\mathcal{B}(p)$, which is the set of elements in the mesh that are incident upon p.

In order not to create skinny elements close to the constrained segments and faces, sequential Delaunay algorithms observe special *encroachment* rules [70, 75–77]. In particular, if a Steiner point p is considered for insertion but it lies within the open equatorial disk of a constrained subfacet f, p is not inserted but the circumcenter of f is inserted instead. Similarly, if p is inside the open diametral circle of a constrained subsegment s,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

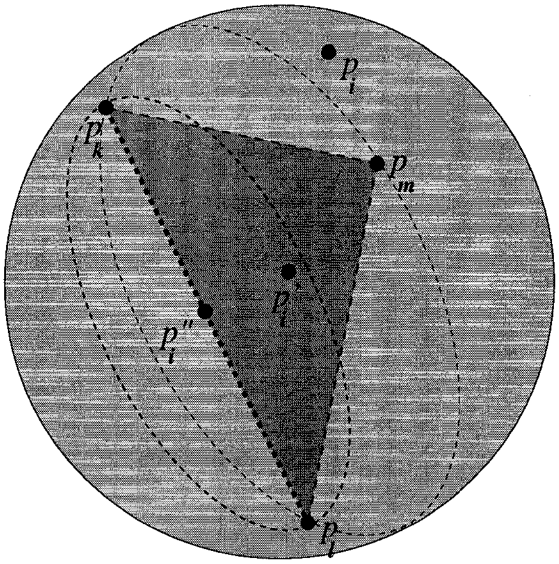


Figure 2.1: Encroachment in three dimensions.

then the midpoint of s is inserted instead. Consider the example in Figure 2.1. The new point p_i is inside the three-dimensional equatorial disk of a constrained face $\triangle(p_k p_l p_m)$. In this case, p_i is rejected and the algorithm attempts to insert the circumcenter p_i' of $\triangle(p_k p_l p_m)$. If p_i' does not encroach upon any constrained segments, it is inserted into the mesh. If, however, it encroaches upon a constrained segment, which is $e(p_k p_l)$ in our example, p_i' is also rejected and the midpoint p_i'' of the constrained edge is inserted.

Definition 2.1 (Local feature size [70, 76, 77]) *The* local feature size *function* $\operatorname{lfs}(p)$ *for a given point* p *is equal to the radius of the smallest disk centered at* p *that intersects two non-incident elements of the PSLG (PLC).*

$\operatorname{lfs}(p)$ satisfies the Lipschitz condition:

Lemma 2.1 (Lemma 1 in [70], Lemma 2 in [77], Lemma 2 in [76]) *Given any PSLG (PLC) and any two points* p_i *and* p_j*, the following inequality holds:*

$$\operatorname{lfs}(p_i) \leq \operatorname{lfs}(p_j) + \|p_i - p_j\|. \tag{2.3}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Here and below we use the standard Euclidean norm $\|\cdot\|$.

Traditionally, Steiner points are selected at the circumcenters of poor quality elements [24, 37, 70, 76, 77]. However, Chew [25] chooses Steiner points randomly inside a *picking sphere* to avoid slivers. The meshes in [25] have constant density; therefore, Chew proves the termination, but not the good grading. Ruppert [70] and Shewchuk [77] proved that if $\bar{\rho} \geq \sqrt{2}$ in two dimensions and $\bar{\rho} \geq 2$ in three dimensions, then Delaunay refinement with circumcenters terminates. If, furthermore, the inequalities are strict, then good grading can also be proven both in two and in three dimensions. In two dimensions, in addition to good grading, one can also prove size-optimality which refers to the fact that the number of triangles in the resulting mesh will be within a constant multiple of the smallest possible number of triangles satisfying the input constraints.

Recently, Üngör [86] introduced a new type of Steiner points called *off-centers.* The idea is based on the observation that sometimes the elements created as a result of inserting circumcenters of skinny triangles are also skinny and require further refinement. This technique combines the advantages of advancing front methods, which can produce very well-shaped elements in practice, and Delaunay methods, which offer theoretical guarantees. Üngör showed that the use of off-centers allows to significantly decrease the size of the final mesh in practice. While eliminating additional point insertions, this strategy creates triangles with the longest possible edges, such that one can prove the termination of the algorithm and still produce a graded mesh.

However, circumcenters and off-centers are not the only possible positions for the insertion of Steiner points, either sequentially or in parallel, such that one can keep the theoretical guarantees. Below we define two types of selection disks: Type I and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Type II. The selection disk of Type II is always inside the selection disk of Type I. Both types of selection disks are defined such that a Delaunay refinement algorithm can pick any point inside a selection disk of a skinny triangle for the elimination of this triangle. First, in Sections 2.2.1 and 2.3.1 we prove that the algorithms which use selection disks of Type I terminate (and therefore the algorithms which use selection disks of Type II also terminate). Then, in Sections 2.2.3 and 2.3.3, we prove that in addition to termination, Delaunay refinement with Type II selection disks also produces theoretically well-graded meshes which are size-optimal in two dimensions.

The traditional proofs of termination and size optimality of Delaunay refinement algorithms [70, 77] explore the relationships between the insertion radius of a point and that of its parent. Stated shortly, the *insertion radius* of point p is the length of the shortest edge connected to p immediately after p is inserted into the mesh, and the *parent* is the vertex which is "responsible" for the insertion of p [77]. The proofs in [70, 77] rely on the assumption that the insertion radius of a Steiner point is equal to the circumradius of the poor quality element. This assumption holds when the Steiner point is chosen at the circumcenter of the element, since by Delaunay property the circumdisk of the element is empty, and, hence, there is no vertex closer to the circumcenter than its vertices. However, the above assumption does not hold if we pick an arbitrary point inside the selection disk of the element. For example, in Figure 2.2(right) the Steiner point p_i within the selection disk can be closer to the mesh vertex p_n than to any of the vertices p_k, p_l, p_m which define the skinny triangle. Therefore, we need to extend the existing theory in the new context, i.e., Steiner points can be inserted anywhere within the selection disks of poor quality elements. The following definitions of insertion radius

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and parent of a Steiner point play a central role in the analysis in [70, 76, 77] and we will adopt them for our analysis, too.

Definition 2.2 (Insertion radius [76, 77]) *The* insertion radius $R(p)$ *of point* p *is the length of the shortest edge which would be connected to* p *if* p *is inserted into the mesh, immediately after it is inserted. If* p *is an input vertex, then* $R(p)$ *is the Euclidean distance between* p *and the nearest input vertex visible from* p.

Remark 2.1 *If* p *is a midpoint of an encroached subsegment or subfacet, then* $R(p)$ *is the distance between* p *and the nearest encroaching mesh vertex; if the encroaching mesh vertex was rejected for insertion, then* $R(p)$ *is the radius of the diametral circle of the subsegment or of the equatorial sphere of the subfacet [76, 77].*

Remark 2.2 *As shown in [76, 77], if* p *is an input vertex, then* $R(p) \geq \text{lfs}(p)$. *Indeed, from the definition of* $\text{lfs}(p)$, *the second feature (in addition to* p*) which intersects the disk centered at* p *is either a constrained segment, a constrained facet, or the nearest input vertex visible from* p.

The following definition of a parent vertex generalizes the corresponding definition in [76, 77]. In our analysis, even though the child is not necessarily the circumcenter, the parent is still defined to be the same vertex.

Definition 2.3 (Parent of a Steiner point) *The* parent $\hat{p}$ *of point* p *is the vertex which is defined as follows:*

- *If* p *is an input vertex, it has no parent.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- *If p is a midpoint of an encroached subsegment or subfacet, then $\hat{p}$ is the encroaching point (possibly rejected for insertion).*
- *If p is inserted inside the circumdisk of a poor quality element (triangle or tetrahedron), $\hat{p}$ is the most recently inserted vertex of the shortest edge of this element.*

2.2 Two-Dimensional Generalized Delaunay Refinement

In this section we introduce two types of selection disks which can be used for the insertion of Steiner points in two dimensions. First, we prove the termination of a Delaunay refinement algorithm with the Type I selection disks. Then we give an example of an optimization based strategy for the insertion of Steiner points from the Type I selection disks which, for small angle bounds, allows to decrease the size of the final mesh in practice. Finally, we introduce the Type II selection disk (which is always inside the Type I selection disk of the same skinny tetrahedron) and prove the good grading and the size optimality.

2.2.1 Proof of Termination with Selection Disks of Type I

Definition 2.4 (Selection disk of Type I in 2D) *If t is a poor quality triangle with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$, then the* Type I selection disk *for the insertion of a Steiner point that would eliminate t is the open disk with center c and radius $r - \sqrt{2}l$.*

For example, in Figure 2.2(right), $e\,(p_l p_m)$ is the shortest edge of a skinny triangle $\triangle\,(p_k p_l p_m)$ and c is its circumcenter. The selection disk of Type I is the shaded disk

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

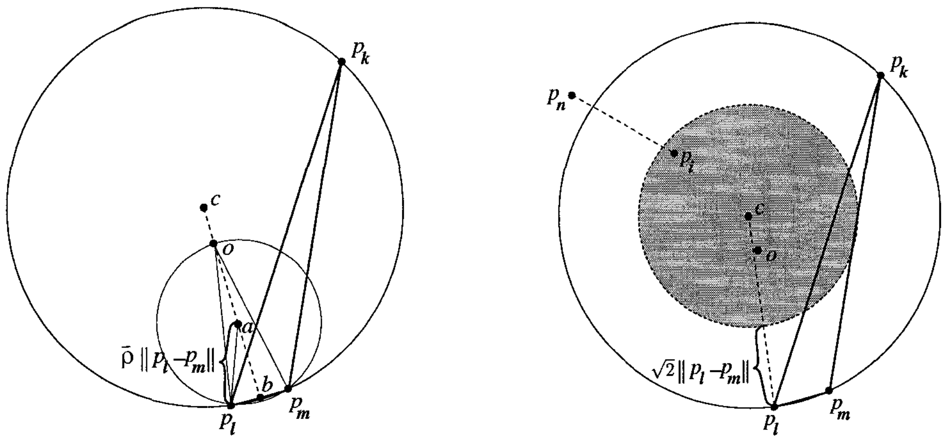


Figure 2.2: **(Left)** Delaunay refinement with the off-centers. [86] **(Right)** The Type I selection disk (shaded) for the insertion of a Steiner point.

with center c and radius $r\left(\triangle\left(p_k p_l p_m\right)\right) - \sqrt{2}\|p_l - p_m\|$.

Further we will prove that any point inside the Type I selection disk of a triangle can be chosen for the elimination of the triangle, and that the generalized Delaunay refinement algorithm which chooses Steiner points inside Type I selection disks terminates.

Remark 2.3 *The radius of Chew's picking sphere is fixed and is equal to one half of the length of the shortest possible edge in the final mesh [25]. We generalize the idea of the picking sphere to the Type I selection disk, such that the radius of the selection disk varies among the triangles and depends on the length of the shortest edge of each particular triangle.*

Remark 2.4 *Üngör's off-center always lies inside the selection disk of Type I. Consider Figure 2.2(left). Suppose $\triangle\left(p_k p_l p_m\right)$ is skinny: $\rho\left(\triangle\left(p_k p_l p_m\right)\right) > \bar{\rho}$. If we insert its circumcenter c, the new triangle $\triangle\left(c p_l p_m\right)$ may also be skinny. In this case, instead of inserting c, Üngör suggests to insert the* off-center *o chosen on the perpendicular*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

bisector of the shortest edge $e\left(p_l p_m\right)$ *in such a way that the new triangle* $\triangle\left(o p_l p_m\right)$ *will have circumradius-to-shortest edge ratio equal to exactly* $\bar{\rho}$, *i.e.,*

$$\rho\left(\triangle\left(o p_l p_m\right)\right)=\bar{\rho}. \tag{2.4}$$

If a *is the circumcenter of* $\triangle\left(o p_l p_m\right)$, *and* b *is the midpoint of edge* $e\left(p_l p_m\right)$, *then from* (2.4) *and the Pythagorean theorem for* $\triangle\left(a b p_l\right)$ *we have:*

$$\|a-b\|^2=\bar{\rho}^2\|p_l-p_m\|^2-\frac{1}{4}\|p_l-p_m\|^2=(\bar{\rho}^2-\frac{1}{4})\|p_l-p_m\|^2,$$

or

$$\|a-b\|=\frac{\sqrt{4\bar{\rho}^2-1}}{2}\|p_l-p_m\|. \tag{2.5}$$

Noting that

$$\|a-o\|=r\left(\triangle\left(o p_l p_m\right)\right)=\bar{\rho}\|p_l-p_m\|, \tag{2.6}$$

we have:

$$\begin{aligned}
\|c-o\| &< r\left(\triangle\left(p_k p_l p_m\right)\right)-\|a-b\|-\|a-o\| \\
&= r\left(\triangle\left(p_k p_l p_m\right)\right)-\tfrac{\sqrt{4\bar{\rho}^2-1}}{2}\|p_l-p_m\|-\bar{\rho}\|p_l-p_m\| && \text{(from (2.5) and (2.6))} \\
&= r\left(\triangle\left(p_k p_l p_m\right)\right)-\left(\tfrac{\sqrt{4\bar{\rho}^2-1}}{2}+\bar{\rho}\right)\|p_l-p_m\| \\
&\leq r\left(\triangle\left(p_k p_l p_m\right)\right)-\left(\tfrac{\sqrt{7}}{2}+\sqrt{2}\right)\|p_l-p_m\| && \text{(since } \bar{\rho}\geq\sqrt{2}\text{)} \\
&< r\left(\triangle\left(p_k p_l p_m\right)\right)-\sqrt{2}\|p_l-p_m\|,
\end{aligned}$$

which implies that the off-center o *is inside the Type I selection disk of triangle* $\triangle\left(p_k p_l p_m\right)$.

Remark 2.5 *As* $\rho\left(\triangle\left(p_k p_l p_m\right)\right)$ *approaches* $\sqrt{2}$, *the Type I selection disk shrinks to the circumcenter* c *of the triangle. If, furthermore,* $\rho\left(\triangle\left(p_k p_l p_m\right)\right)\leq\sqrt{2}$, *the selection disk vanishes, which coincides with the fact that the triangle* $\triangle\left(p_k p_l p_m\right)$ *cannot be considered skinny.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Lemma 2.2 *If p_i is a vertex of the mesh produced by a Delaunay refinement algorithm which chooses points within Type I selection disks of triangles with circumradius-to-shortest edge ratios greater than $\bar{\rho} \geq \sqrt{2}$, then the following inequality holds:*

$$R(p_i) \geq C \cdot R(\hat{p}_i), \tag{2.7}$$

where C is defined as follows:

(i) $C = \sqrt{2}$ if p_i is a Steiner point chosen within the Type I selection disk of a skinny triangle;

Otherwise, let p_i be the midpoint of subsegment s. Then

(ii) $C = \frac{1}{\sqrt{2}}$ if $\hat{p}_i$ is a Steiner point which encroaches upon s, chosen within the selection disk of a skinny triangle;

(iii) $C = \frac{1}{2\cos\alpha}$ if p_i and $\hat{p}_i$ lie on incident subsegments separated by an angle of α (with $\hat{p}_i$ encroaching upon s), where $45° \leq \alpha \leq 90°$;

(iv) $C = \sin\alpha$ if p_i and $\hat{p}_i$ lie on incident segments separated by an angle of $\alpha \leq 45°$.

If p_i is an input vertex, then

$$R(p_i) \geq \mathrm{lfs}(p_i). \tag{2.8}$$

Proof: We need to present new proofs only for cases (i) and (ii), since the proofs for all other cases are independent of the choice of the point within the selection disk and are given in [77].

Case (i) By the definition of a parent vertex, $\hat{p}_i$ is the most recently inserted endpoint of the shortest edge of the triangle; without loss of generality let $\hat{p}_i = p_l$ and $e(p_l p_m)$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

be the shortest edge of the skinny triangle $\triangle(p_k p_l p_m)$, see Figure 2.2(right). If $e(p_l p_m)$ was the shortest edge among the edges incident upon p_l at the time p_l was inserted into the mesh, then $\|p_l - p_m\| = R(p_l)$ by the definition of the insertion radius; otherwise, $\|p_l - p_m\| \geq R(p_l)$. In either case,

$$\|p_l - p_m\| \geq R(p_l). \tag{2.9}$$

Now we can derive the relation between the insertion radius of point p_i and the insertion radius of its parent $\hat{p}_i = p_l$:

$$\begin{array}{rcll} R(p_i) & > & \sqrt{2}\|p_l - p_m\| & \text{(from Delaunay property and Definition 2.4)} \\ & \geq & \sqrt{2}R(p_l) & \text{(from (2.9)).} \end{array}$$

Hence, $R(p_i) > \sqrt{2}R(\hat{p}_i)$; choose $C = \sqrt{2}$.

Case (ii) Let $\hat{p}_i$ be inside the selection disk of a skinny triangle $\triangle(p_k p_l p_m)$, such that $\hat{p}_i$ encroaches upon $e(p_u p_v)$, see Figure 2.4(left). Since the edge $e(p_u p_v)$ is part of the mesh, there must exist some vertex p_w such that p_u, p_v, and p_w form a triangle. Because p_w is outside of the diametral circle of $e(p_u p_v)$, the circumdisk $\bigcirc(\triangle(p_u p_v p_w))$ has to include point $\hat{p}_i$. Therefore, if $\hat{p}_i$ were inserted into the mesh, $\triangle(p_u p_v p_w)$ would be part of the cavity $\mathcal{C}(\hat{p}_i)$ and the edges connecting $\hat{p}_i$ with p_u and p_v would be created. Therefore,

$$\begin{array}{rcll} R(\hat{p}_i) & \leq & \min(\|\hat{p}_i - p_u\|, \|\hat{p}_i - p_v\|) & \text{(from the definition of insertion radius)} \\ & < & \sqrt{2}\frac{\|p_u - p_v\|}{2} & \text{(because } \hat{p}_i \text{ encroaches upon } e(p_u p_v)\text{)} \\ & = & \sqrt{2}R(p_i) & \text{(from the definition of insertion radius);} \end{array}$$

choose $C = \frac{1}{\sqrt{2}}$. ■

Figure 2.3 shows the relationship between the insertion radii of mesh vertices and the insertion radii of their parents. We can see that if Inequality (2.7) is satisfied then

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

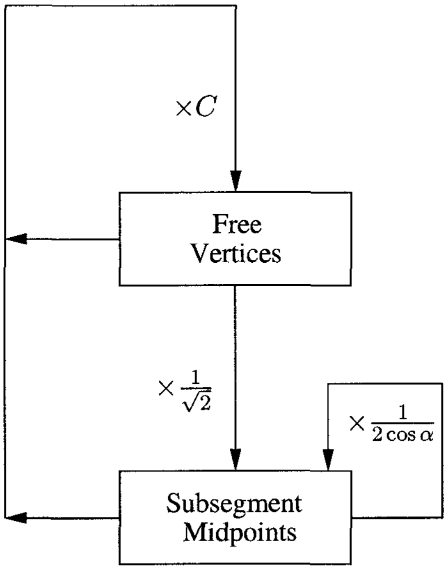


Figure 2.3: Flow diagram from [77] illustrating the relationship between the insertion radius of a vertex and its parent in two dimensions. If no cycle has a product smaller then one, the algorithm will terminate. Input vertices are not shown since they do not participate in cycles. In [77] the constant $C = \bar{\rho} \geq \sqrt{2}$. In our case, with the use of Type I selection disks $C = \sqrt{2}$, and with the use of Type II selection disks $C = \delta\bar{\rho} \geq \sqrt{2}$.

no new edge will be created whose length is smaller than $\frac{1}{\sqrt{2}}$ times the length of some existing edge and the algorithm will eventually terminate because it will run out of space to insert new vertices.

Theorem 2.1 (Theorem 4 in [77]) *Let* $\mathrm{lfs}_{\min}$ *be the shortest distance between two non-incident entities (vertices or segments) of the input PSLG. Suppose that any two incident segments are separated by an angle of at least* $60°$*, and a triangle is considered to be skinny if its circumradius-to-shortest edge ratio is larger than* $\bar{\rho}$*, where* $\bar{\rho} \geq \sqrt{2}$*. Ruppert's algorithm will terminate with no triangulation edge shorter than* $\mathrm{lfs}_{\min}$.

The proof of this theorem in [77] is based on the result of Lemma 3 in [77], which establishes the inequality (2.7) in the context of circumcenter point insertion. Otherwise, the proof is independent of the specific position of inserted points. Since we proved (2.7)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in the context of inserting arbitrary points within Type I selection disks, this theorem also holds in this new context.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

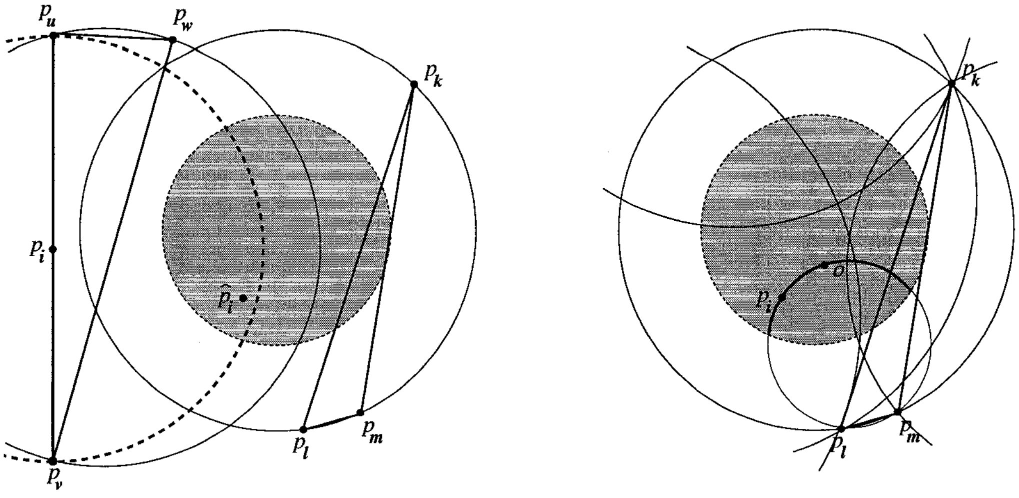


Figure 2.4: **(Left)** $\hat{p}_i$ is a Steiner point within a selection disk of a poor quality triangle which encroaches upon a constrained segment $e\,(p_u p_v)$. **(Right)** An example of an optimization-based method for the selection of a Steiner point within a selection disk of a poor quality triangle.

2.2.2 An Example of a Point Selection Strategy

Let us consider Figure 2.4(right). We can see that the off-center o of the skinny triangle $\triangle\,(p_k p_l p_m)$ is not the only location for a Steiner point p_i that will lead to the creation of the new triangle incident to the edge $e\,(p_l p_m)$ with circumradius-to-shortest edge ratio equal to exactly $\bar{\rho}$. The arc shown in bold in the figure is the intersection of the circle passing through p_l, p_m, and o with the selection disk of $\triangle\,(p_k p_l p_m)$. Let us denote this arc as Γ. The thin arcs show parts of the circumcircles of other triangles in the mesh. We can observe that the cavity $\mathcal{C}\,(p_i)$ will vary depending on the location of p_i, according to the set of triangle circumdisks that include p_i. Let us also represent the penalty for deleting triangle t as $P(t)$:

$$P(t) = \begin{cases} -1, & \text{if } (\rho\,(t) > \bar{\rho}) \vee (\mathcal{A}\,(t) > \bar{\mathcal{A}}), \\ 1, & \text{otherwise.} \end{cases}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Then our objective is to minimize the profit function associated with the insertion of point p_i as the sum of the penalties for deleting all triangles in the cavity $\mathcal{C}(p_i)$:

$$\min_{p_i \in \Gamma} F(p_i), \quad F(p_i) = \sum_{t \in \mathcal{C}(p_i)} P(t)$$

In other words, we try to minimize the number of deleted good quality triangles and at the same time to maximize the number of deleted poor quality triangles. The results of our experiments with the cylinder flow (Figure 2.5(left)) and the pipe cross-section (Figure 2.5(right)) models using `Triangle` version 1.6 [73] are summarized in Tables 2.1 and 2.2. We do not list the running times since our experimental implementation is built on top of `Triangle`, but we do not take advantage of its efficient routines for our intermediate calculations as do the circumcenter and the off-center point insertion methods. From Table 2.1 we can see that our optimization-based method tends to produce up to 20% fewer triangles than the circumcenter method and up to 5% fewer triangles than the off-center method for small values of the minimal angle bound and no area bound, and the improvement diminishes as the angle bound increases. Parts of the pipe mesh for the three point insertion methods are also shown in Figure 2.6. Table 2.2 lists the results of the similar experiments, with an additional area bound constraint. We observe that the introduction of an area bound effectively voids the difference among the presented point insertion strategies.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

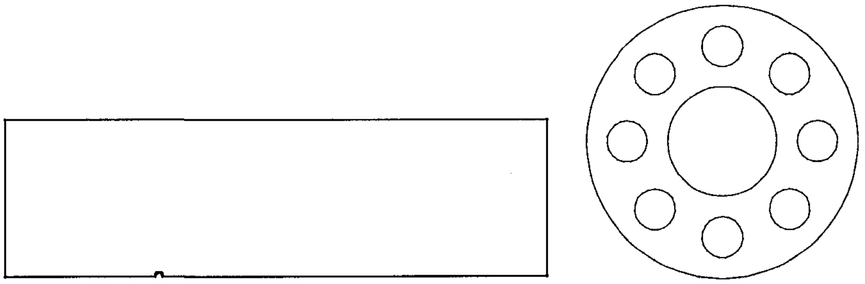


Figure 2.5: **(Left)** An upper part of a model of cylinder flow. **(Right)** Pipe cross-section model.

Table 2.1: Mesh quality comparison for three point insertion strategies, no area bound is used. θ is the minimal angle bound, n is the number of triangles in the resulting mesh, % is the percentage ratio in the number of triangles over the optimization-based method, $\min A$ and $\max A$ are the minimal and the maximal angles in the entire mesh, A^{ave}_{min} and A^{ave}_{max} are the averages of the minimum and the maximum angle for all triangles.

Point position		$\theta = 10^\circ$		$\theta = 20^\circ$		$\theta = 30^\circ$	
		flow	pipe	flow	pipe	flow	pipe
Circumcenter	n	2173	3033	3153	4651	8758	10655
	%	120.3	106.7	107.5	106.6	138.5	124.1
	$\min A$	10.0	10.0	20.0	20.0	30.0	30.0
	$\max A$	151.5	150.4	128.5	137.7	119.3	119.4
	A^{ave}_{min}	28.1	26.7	35.9	35.9	45.8	45.6
	A^{ave}_{max}	92.5	92.5	88.1	88.0	76.6	76.8
Off-center	n	1906	2941	2942	4411	6175	8585
	%	105.5	103.5	100.3	101.1	97.7	100.0
	$\min A$	10.1	10.0	20.2	20.0	30.0	30.0
	$\max A$	157.0	149.9	133.3	133.7	118.2	119.0
	A^{ave}_{min}	24.4	25.4	34.4	34.7	43.6	43.7
	A^{ave}_{max}	96.2	94.9	87.8	87.1	77.3	77.7
Our example of an optimization-based method	n	1805	2841	2932	4359	6319	8581
	%	100.0	100.0	100.0	100.0	100.0	100.0
	$\min A$	10.0	10.0	20.0	20.0	30.0	30.0
	$\max A$	157.3	152.5	138.3	137.1	119.0	119.5
	A^{ave}_{min}	23.2	24.8	34.3	34.6	44.0	43.6
	A^{ave}_{max}	98.3	96.7	87.9	87.6	77.1	77.8

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

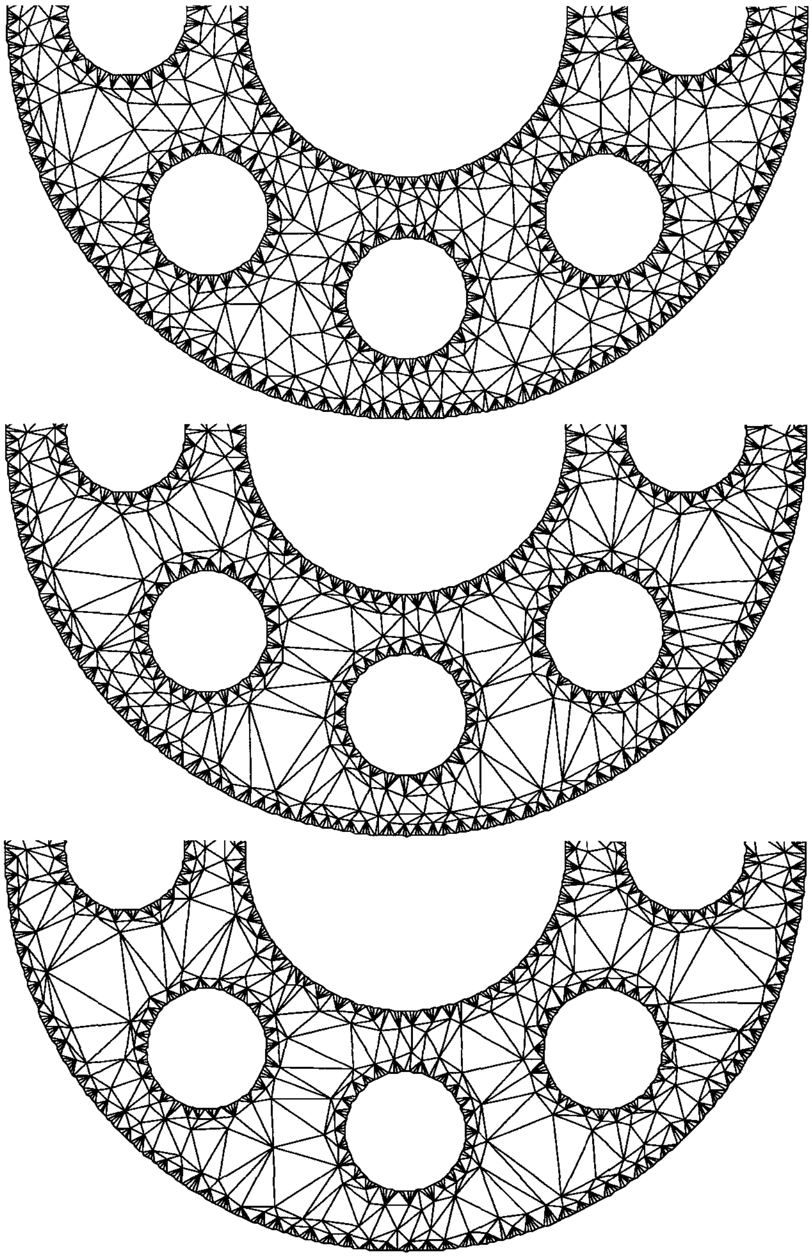


Figure 2.6: The pipe mesh (lower half is shown) with the minimal angle bound equal to 10°. **(Top)** Steiner points are inserted at the circumcenters of skinny triangles: 3033 triangles. **(Center)** Steiner points are inserted at the off-centers: 2941 triangles. **(Bottom)** Our optimization-based method: 2841 triangles.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2.2: The comparison of the number of triangles generated with the use of circumcenter, off-center, and an optimization-based point insertion strategies, with an area bound. For the cylinder flow model, the area bound is set to $\bar{\mathcal{A}} = 0.01$, and for the pipe cross-section model $\bar{\mathcal{A}} = 1.0$.

Point position	$\theta = 10°$		$\theta = 20°$		$\theta = 30°$	
	flow	pipe	flow	pipe	flow	pipe
Circumcenter	219914	290063	220509	289511	228957	294272
Off-center	219517	290057	220479	289331	226894	295644
Our example of an optimization-based method	219470	289505	220281	289396	226585	294694

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.2.3 Proof of Good Grading and Size Optimality with Selection Disks of Type II

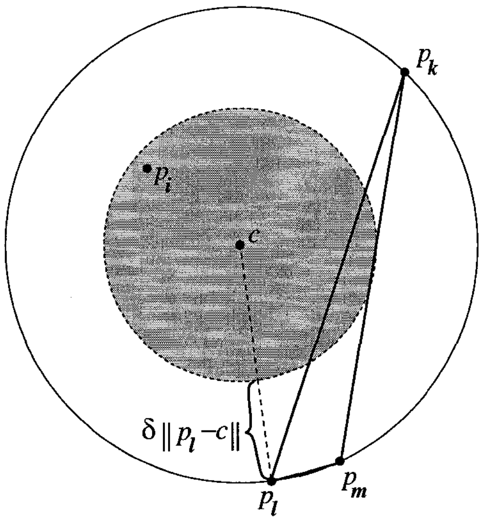


Figure 2.7: Selection of a Steiner point by a δ-graded Delaunay refinement algorithm. When $\delta = \sqrt{2}/\bar{\rho}$ the shaded disk corresponds to the Type II selection disk.

Definition 2.5 (Selection disk of Type II in 2D) *If t is a poor quality triangle with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$, then the* Type II selection disk *for the insertion of a Steiner point that would eliminate t is the open disk with center c and radius $r(1 - \frac{\sqrt{2}}{\bar{\rho}})$.*

For example, in Figure 2.7, $e\,(p_l p_m)$ is the shortest edge of a skinny triangle $\triangle\,(p_k p_l p_m)$ and c is its circumcenter. The Type II selection disk for this triangle is the shaded disk with center c and radius $r\,(\triangle\,(p_k p_l p_m))\,(1 - \frac{\sqrt{2}}{\bar{\rho}})$.

Remark 2.6 *Note from Definitions 2.4 and 2.5 that the radius of the Type II selection disk is always smaller than the radius of the Type I selection disk of the same skinny triangle because $r(1 - \frac{\sqrt{2}}{\bar{\rho}}) = r - \frac{\rho}{\bar{\rho}}\sqrt{2}l$ and $\rho > \bar{\rho}$.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Remark 2.7 *As $\bar{\rho}$ approaches $\sqrt{2}$ the radius of the Type II selection disk approaches zero, which means that the selection disk shrinks to the circumcenter point.*

As can be seen further below, the price which we pay for the gain in the flexibility in choosing points is the increase of the constants which bound the size of the mesh. To classify the Delaunay refinement algorithms with respect to the theoretical bounds on mesh grading we need the following definition.

Definition 2.6 (δ-graded Delaunay refinement algorithm in 2D) *If for every triangle t with circumcenter c, circumradius r, shortest edge length l, and circumradius-to-shortest edge length ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$ a Delaunay refinement algorithm selects a Steiner point p_i within the Type II selection disk such that $\|p_i - c\| < r(1-\delta)$, where*

$$\frac{\sqrt{2}}{\bar{\rho}} \leq \delta \leq 1,$$

we say that this Delaunay refinement algorithm is δ-graded.

The analysis below assumes that all angles in the input PSLG are greater than 45° or 60°. In practice such geometries are rare; however, a modification of the algorithm with a concentric circular shell splitting [70, 77] allows to guarantee the termination of the algorithm even though the small angles adjacent to the segments of the input PSLG cannot be improved.

Lemma 2.3 *If p_i is a vertex of the mesh produced by a δ-graded Delaunay refinement algorithm then the following inequality holds:*

$$R(p_i) \geq C \cdot R(\hat{p}_i), \tag{2.10}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where we distinguish among the following cases:

(i) $C = \delta\bar{\rho}$ *if* p_i *is a Steiner point chosen within the Type II selection disk of a skinny triangle;*

Otherwise, let p_i *be the midpoint of subsegment s. Then*

(ii) $C = \frac{1}{\sqrt{2}}$ *if* $\hat{p}_i$ *is a Steiner point which encroaches upon s, chosen within the Type II selection disk of a skinny triangle;*

(iii) $C = \frac{1}{2\cos\alpha}$ *if* p_i *and* $\hat{p}_i$ *lie on incident subsegments separated by an angle of* α *(with* $\hat{p}_i$ *encroaching upon s), where* $45^\circ \leq \alpha \leq 90^\circ$*;*

(iv) $C = \sin\alpha$ *if* p_i *and* $\hat{p}_i$ *lie on incident segments separated by an angle of* $\alpha \leq 45^\circ$*.*

If p_i *is an input vertex, then*

$$R(p_i) \geq \operatorname{lfs}(p_i).$$

Proof: We need to present a new proof only for case (i) since the proof for case (ii) is the same as in Lemma 2.2, and the proofs for all other cases are independent of the choice of the point within the selection disk and are given in [77].

Case (i) As in the proof of Case (i) of Lemma 2.2, assuming that $e(p_l p_m)$ is the shortest edge of the skinny triangle $\triangle(p_k p_l p_m)$ and $\hat{p}_i = p_l$, we derive relation (2.9). Then

$$\begin{aligned}
R(p_i) &> \delta r && \text{(from Delaunay property and Definition 2.6)}\\
&= \delta\rho \|p_l - p_m\| && \text{(since } \rho = \tfrac{r}{\|p_l - p_m\|}\text{)}\\
&> \delta\bar{\rho}\|p_l - p_m\| && \text{(since } \rho > \bar{\rho}\text{)}\\
&\geq \delta\bar{\rho} R(p_l) && \text{(from (2.9)).}
\end{aligned}$$

Hence, $R(p_i) > \delta\bar{\rho}R(\hat{p}_i)$; choose $C = \delta\bar{\rho}$. ■

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The quantity $D(p)$ is defined as the ratio of $\operatorname{lfs}(p)$ over $R(p)$ [77]:

$$D(p) = \frac{\operatorname{lfs}(p)}{R(p)}. \tag{2.11}$$

It reflects the density of vertices near p at the time p is inserted, weighted by the local feature size.

Lemma 2.4 *If p is a vertex of the mesh produced by a δ-graded Delaunay refinement algorithm and C is the constant specified by Lemma 2.3, then the following inequality holds:*

$$D(p) \leq \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C}. \tag{2.12}$$

Proof: If p is inside a Type II selection disk of a skinny triangle with circumradius r, then

$$\begin{aligned} \|p-\hat{p}\| &< 2r - \delta r && \text{(from the definition of } \hat{p} \text{ and Def. 2.6)} \\ &= (2-\delta)r && \\ &= \tfrac{2-\delta}{\delta}\delta r && \\ &< \tfrac{2-\delta}{\delta} R(p) && \text{(from Delaunay property and Def. 2.6).} \end{aligned}$$

If p is an input vertex or lies on an encroached segment, then

$$\begin{aligned} \|p-\hat{p}\| &\leq R(p) && \text{(by definitions of } \hat{p} \text{ and } R(p)\text{)} \\ &\leq \tfrac{2-\delta}{\delta} R(p) && \text{(since from Def. 2.6, } \delta \leq 1\text{).} \end{aligned}$$

In all cases,

$$\|p-\hat{p}\| \leq \frac{2-\delta}{\delta} R(p). \tag{2.13}$$

Then

$$\begin{aligned} \operatorname{lfs}(p) &\leq \operatorname{lfs}(\hat{p}) + \|p-\hat{p}\| && \text{(from Lemma 2.1)} \\ &\leq \operatorname{lfs}(\hat{p}) + \tfrac{2-\delta}{\delta} R(p) && \text{(from (2.13))} \\ &= D(\hat{p}) R(\hat{p}) + \tfrac{2-\delta}{\delta} R(p) && \text{(from (2.11))} \\ &\leq D(\hat{p}) \tfrac{R(p)}{C} + \tfrac{2-\delta}{\delta} R(p) && \text{(from Lemma 2.3).} \end{aligned}$$

The result follows from the division of both sides by $R(p)$. ■

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Lemma 2.5 (Extension of Lemma 7 in [77] and Lemma 2 in [70]) *Suppose that* $\bar{\rho} > \sqrt{2}$ *and the smallest angle in the input PSLG is strictly greater than* $60°$. *There exist fixed constants* C_T *and* C_S *such that, for any vertex* p *inserted (or considered for insertion and rejected) by a* δ*-graded Delaunay refinement algorithm,* $D(p) \leq C_T$, *and for any vertex* p *inserted at the midpoint of an encroached subsegment,* $D(p) \leq C_S$. *Hence, the insertion radius of a vertex has a lower bound proportional to its local feature size.*

Proof: The proof is by induction and is similar to the proof of Lemma 7 in [77]. The base case covers the input vertices, and the inductive step covers the other two types of vertices: free vertices and subsegment midpoints.

Base case: The lemma is true if p is an input vertex, because in this case, by Remark 2.2,

$$D(p) = \frac{\mathrm{lfs}(p)}{R(p)} \leq 1.$$

Inductive hypothesis: Assume that the lemma is true for $\hat{p}$, i.e.,

$$D(\hat{p}) \leq \max\{C_T, C_S\}.$$

Inductive step: There are two cases:

(i) If p is in the Type II selection disk of a skinny triangle, then

$$\begin{aligned} D(p) &\leq \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{C} && \text{(from Lemma 2.4)} \\ &= \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{\delta\bar{\rho}} && \text{(from Lemma 2.3)} \\ &\leq \tfrac{2-\delta}{\delta} + \tfrac{\max\{C_T,C_S\}}{\delta\bar{\rho}} && \text{(by the inductive hypothesis).} \end{aligned}$$

It follows that one can prove that $D(p) \leq C_T$ if C_T is chosen so that

$$\frac{2-\delta}{\delta} + \frac{\max\{C_T, C_S\}}{\delta\bar{\rho}} \leq C_T. \tag{2.14}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(ii) If p is the midpoint of a subsegment s such that $\hat{p}$ encroaches upon s, then we have 3 sub-cases:

(ii-a) If $\hat{p}$ is an input vertex, then the disk centered at p and touching $\hat{p}$ has radius less than the radius of the diametral disk of s and therefore $\operatorname{lfs}(p) < R(p)$. Thus, $D(p) < 1$ and the lemma holds.

(ii-b) If $\hat{p}$ is a rejected point from the Type II selection disk of a skinny triangle or lies on a segment not incident to s, then

$$\begin{array}{rcll} D(p) & \leq & \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C} & \text{(from Lemma 2.4)} \\ & = & \frac{2-\delta}{\delta} + \sqrt{2}D(\hat{p}) & \text{(from Lemma 2.3)} \\ & \leq & \frac{2-\delta}{\delta} + \sqrt{2}C_T & \text{(by the inductive hypothesis).} \end{array}$$

(ii-c) If $\hat{p}$ lies on a segment incident to s, then

$$\begin{array}{rcll} D(p) & \leq & \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C} & \text{(from Lemma 2.4)} \\ & = & \frac{2-\delta}{\delta} + 2\cos\alpha D(\hat{p}) & \text{(from Lemma 2.3)} \\ & \leq & \frac{2-\delta}{\delta} + 2C_S\cos\alpha & \text{(by the inductive hypothesis).} \end{array}$$

It follows that one can prove that $D(p) \leq C_S$ if C_S is chosen so that both of the following relations (2.15) and (2.16) are satisfied:

$$\frac{2-\delta}{\delta} + \sqrt{2}C_T \leq C_S, \tag{2.15}$$

and

$$\frac{2-\delta}{\delta} + 2C_S\cos\alpha \leq C_S. \tag{2.16}$$

If $\delta\bar{\rho} > \sqrt{2}$, relations (2.14) and (2.15) can be simultaneously satisfied by choosing

$$C_T = \frac{(2-\delta)(\bar{\rho}+\delta)}{\delta\bar{\rho}-\sqrt{2}}, \quad \text{and} \quad C_S = \frac{(2-\delta)\bar{\rho}(1+\sqrt{2})}{\delta\bar{\rho}-\sqrt{2}}.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If the smallest input angle $\alpha_{\min} > 60°$, relations (2.14) and (2.16) can be simultaneously satisfied by choosing

$$C_T = \frac{2-\delta}{\delta} + \frac{C_S}{\delta \bar{\rho}}, \quad \text{and} \quad C_S = \frac{2-\delta}{\delta(1-2\cos\alpha_{\min})}.$$

■

Theorem 2.2 (Theorem 8 in [77], Theorem 1 in [70]) *For any vertex p of the output mesh, the distance to its nearest neighbor is at least* $\frac{\operatorname{lfs}(p)}{C_S+1}$.

The proof in [77] relies only on Lemmata 2.1 and 2.5 here and, therefore, holds for the arbitrary points chosen within selection disks of skinny triangles.

Theorem 2.2 is used in the proof of the following theorem.

Theorem 2.3 (Theorem 10 in [77], Theorem 14 in [65], Theorem 3 in [70]) *Let* $\operatorname{lfs}_{\mathcal{M}}(p)$ *be the local feature size at p with respect to a mesh* $\mathcal{M}$ *(treating* $\mathcal{M}$ *as a PSLG), whereas* $\operatorname{lfs}(p)$ *remains the local feature size at p with respect to the input PSLG. Suppose a mesh* $\mathcal{M}$ *with smallest angle* θ *has the property that there is some constant* $k_1 \geq 1$*, such that for every point p,* $k_1 \operatorname{lfs}_{\mathcal{M}}(p) \geq \operatorname{lfs}(p)$*. Then the cardinality of* $\mathcal{M}$ *is less than* k_2 *times the cardinality of any other mesh of the input PSLG with smallest angle* θ*, where* $k_2 = \mathcal{O}\left(k_1^2/\theta\right)$.

Smaller values of δ offer more flexibility to a δ-graded Delaunay refinement algorithm in choosing Steiner points. However, from Lemma 2.5 it follows that as $\delta\bar{\rho}$ approach $\sqrt{2}$, C_T and C_S approach infinity, which leads to the worsening of the good grading guarantees. Therefore, along with satisfying application-specific requirements, the point insertion schemes should try to place Steiner points as close to circumcenters as possible.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 Three-Dimensional Generalized Delaunay Refinement

In this section we introduce two types of selection disks which can be used for the insertion of Steiner points. First, we prove the termination of a Delaunay refinement algorithm with the Type I selection disks. Then we give an example of an optimization based strategy for the insertion of Steiner points from the Type I selection disks. Finally, we introduce the Type II selection disk (which is always inside the Type I selection disk of the same skinny tetrahedron) and prove the good grading. As in [76], the proofs require that all incident segments and faces in the input geometry are separated by angles of at least 90°.

2.3.1 Proof of Termination with Selection Disks of Type I

Definition 2.7 (Selection disk of Type I in 3D) *If τ is a poor quality tetrahedron with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq 2$, then the* selection disk of Type I *for the insertion of a Steiner point that would eliminate τ is the open disk with center c and radius $r - 2l$.*

Following [76], the analysis below requires that the Delaunay refinement algorithm prioritize the splitting of encroached faces. In particular, when a Steiner point p encroaches upon several constrained faces, the encroached face which contains the projection of p is split. The projection of a point p onto a plane is the point in the plane which is closest to p. This requirement allows to achieve better bounds on the circumradius-to-shortest edge ratios in the final mesh.

Lemma 2.6 (Projection Lemma [76]) *Let f be a subfacet of the Delaunay trian-*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

gulated facet F. Suppose that f is encroached upon by some vertex p, but p does not encroach upon any subsegment of F. Then $\mathrm{proj}_F(p)$ *lies in the facet F, and p encroaches upon a subfacet of F that contains* $\mathrm{proj}_F(p)$.

Now we can prove the following lemma which establishes the relationship between the insertion radius of a point and its parent.

Lemma 2.7 *If p_i is a vertex of the mesh produced by a Delaunay refinement algorithm which chooses points within Type I selection disks of tetrahedra with circumradius-to-shortest edge ratios greater than $\bar{\rho} \geq 2$, then the following inequality holds:*

$$R(p_i) \geq C \cdot R(\hat{p}_i), \tag{2.17}$$

where we distinguish among the following cases:

(i) $C = 2$ if p_i is a Steiner point chosen within the Type I selection disk of a skinny tetrahedron;

(ii) $C = \frac{1}{\sqrt{2}}$ if p_i is a circumcenter of an encroached constrained face;

(iii) $C = \frac{1}{\sqrt{2}}$ if p_i is a midpoint of an encroached constrained segment.

If p_i is an input vertex, then

$$R(p_i) \geq \mathrm{lfs}(p_i).$$

Proof: We need to prove only cases (i) and (ii) since the proofs of all other cases are independent of the choice of the point within the selection disk and are given in [76].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

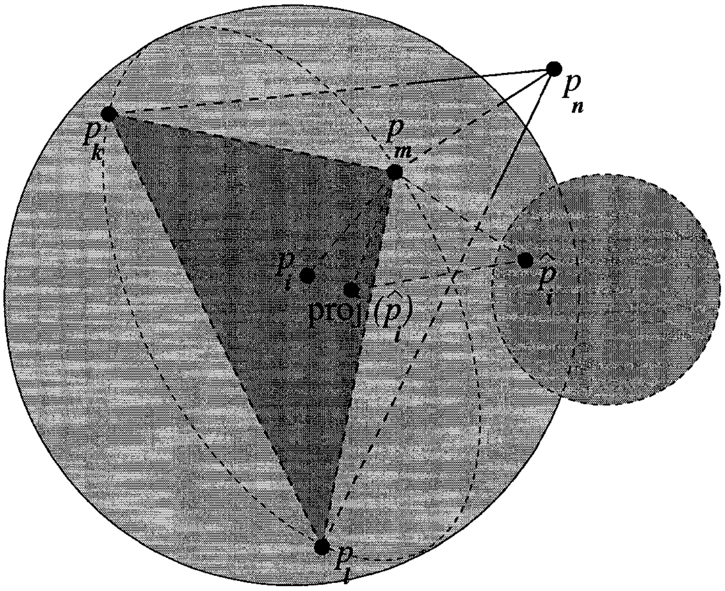


Figure 2.8: An illustration of the relationship between the insertion radii of Steiner points, in the case of encroachment in three dimensions.

Case (i) Without the loss of generality, let $e\,(p_l p_m)$ be the shortest edge of the skinny tetrahedron $\tau\,(p_k p_l p_m p_n)$ and $\hat{p}_i = p_l$. Then

$$\begin{aligned} R\,(p_i) &> 2\|p_l - p_m\| && \text{(from Delaunay property and Definition 2.7)} \\ &\geq 2R\,(p_l) && \text{(from Definition 2.2 of insertion radius)} \\ &= 2R\,(\hat{p}_i)\,; \end{aligned}$$

choose $C = 2$.

Case (ii) Consider Figure 2.8. Let $\hat{p}_i$ be inside the selection disk (the smaller shaded circle) of some skinny tetrahedron (not shown), such that $\hat{p}_i$ encroaches upon the constrained face $\triangle\,(p_k p_l p_m)$, and p_i is the circumcenter of $\triangle\,(p_k p_l p_m)$. According to the projection requirement, let $\triangle\,(p_k p_l p_m)$ include $\mathrm{proj}_F\,(\hat{p}_i)$, where F is the facet of the input PLC containing $\triangle\,(p_k p_l p_m)$. Without the loss of generality, suppose p_m is the point closest to $\mathrm{proj}_F\,(\hat{p}_i)$ among the vertices of $\triangle\,(p_k p_l p_m)$. Because $\mathrm{proj}_F\,(\hat{p}_i)$ lies inside the triangle $\triangle\,(p_k p_l p_m)$, it cannot be father away from p_m than the circumcenter

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of $\triangle (p_k p_l p_m)$, i.e.,

$$\|\mathrm{proj}_F (\hat{p}_i) - p_m\| \le r(\triangle (p_k p_l p_m)). \tag{2.18}$$

Furthermore, because $\hat{p}_i$ is inside the equatorial disk of $\triangle (p_k p_l p_m)$,

$$\|\mathrm{proj}_F (\hat{p}_i) - \hat{p}_i\| < r(\triangle (p_k p_l p_m)). \tag{2.19}$$

From (2.18) and (2.19), as well as the fact that the triangle with vertices $\hat{p}_i$, $\mathrm{proj}_F (\hat{p}_i)$, and p_m has a right angle at $\mathrm{proj}_F (\hat{p}_i)$, we have:

$$\|\hat{p}_i - p_m\| < \sqrt{2} r(\triangle (p_k p_l p_m)). \tag{2.20}$$

Since the face $\triangle (p_k p_l p_m)$ is part of the mesh, there must exist some vertex p_n such that p_k, p_l, p_m, and p_n form a tetrahedron. p_n is outside of the equatorial disk of $\triangle (p_k p_l p_m)$ because all encroachment occurrences are eliminated before the insertion of Steiner points inside circumdisks of poor quality tetrahedra. Therefore, the circumdisk $\bigcirc (\tau (p_k p_l p_m p_n))$ has to include point $\hat{p}_i$. Hence, if $\hat{p}_i$ were inserted into the mesh, $\tau (p_k p_l p_m p_n)$ would be part of the cavity $\mathcal{C}(\hat{p}_i)$ and the edges connecting $\hat{p}_i$ with p_k, p_l, and p_m would be created. Using (2.20), we have:

$$R(\hat{p}_i) \le \|\hat{p}_i - p_m\| < \sqrt{2} r(\triangle (p_k p_l p_m)) = \sqrt{2} R(p_i);$$

choose $C = \frac{1}{\sqrt{2}}$. ■

Figure 2.9 shows the relationship between the insertion radii of mesh vertices and the insertion radii of their parents. We can see that if Inequality 2.17 is satisfied then no new edge will be created whose length is smaller than half of the length of some existing

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

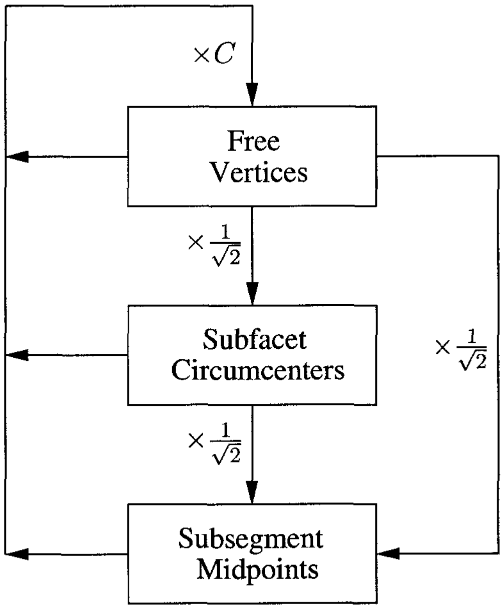

×C
Free Vertices
$\times\frac{1}{\sqrt{2}}$
Subfacet Circumcenters
$\times\frac{1}{\sqrt{2}}$
$\times\frac{1}{\sqrt{2}}$
Subsegment Midpoints

Figure 2.9: Flow diagram from [76] illustrating the relationship between the insertion radius of a vertex and that of its parent in three dimensions. If no cycle has a product smaller then one, the algorithm will terminate. Input vertices are not shown since they do not participate in cycles. In [76] the constant $C = \bar{\rho} \geq 2$. In our case, with the use of Type I selection disks $C = 2$, and with the use of Type II selection disks $C = \delta\bar{\rho} \geq 2$.

edge and the algorithm will eventually terminate because it will run out of places to insert new vertices.

2.3.2 An Example of a Point Selection Strategy

In two dimensions, by selecting the new Steiner point, we can construct only one new triangle which will be incident upon the shortest edge of the existing skinny triangle and which will have the required circumradius-to-shortest edge ratio. The three-dimensional case, however, is complicated by the fact that several new tetrahedra may be incident upon the shortest edge of the existing skinny tetrahedron. The example in Figure 2.10 shows a skinny tetrahedron $\tau\,(p_k p_l p_m p_n)$ with two new tetrahedra $\tau\,(p_i p_k p_l p_n)$ and $\tau\,(p_i p_k p_l p_m)$ that are incident upon the shortest edge $e\,(p_k p_l)$. By having to deal with multiple incident tetrahedra we face a multy-constrained optimization problem

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

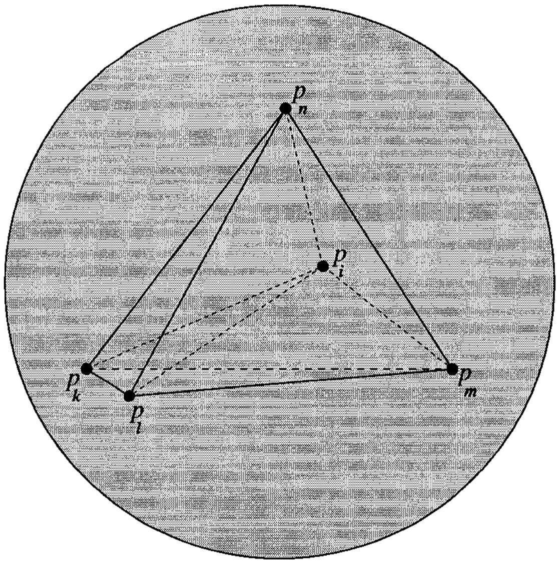


Figure 2.10: A tetrahedron with high circumradius-to-shortest edge ratio.

which to the best of our knowledge has not been analyzed with respect to the existence of the optimal solution and the construction of this solution.

A heuristic approach was suggested by Üngör [87] who proposed two types of off-center points in three dimensions. Based on the experimental data, he observes the following: "Use of both types of off-centers or the use of TYPE II off-centers alone outperforms the use of TYPE I off-centers alone, which in turn outperforms the use of circumcenters." If a is the midpoint of the shortest edge $e\left(p_k p_l\right)$ of the tetrahedron and c is its circumcenter, than the TYPE II off-center b is computed in [87] on the segment $\mathcal{L}\left(ac\right)$ such that

$$|\mathcal{L}\left(ab\right)| = \alpha_3\left(\bar{\rho} + \sqrt{\bar{\rho}^2 - 1/4}\right),$$

where α_3 is the perturbation factor. From experimental evidence in [87] the author suggests that a good choice for α_3 is 0.6.

The insertion of TYPE II off-centers was implemented by Hang Si in `Tetgen` version 1.4.0 along with the circumcenter insertion method. We also added the implementation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2.3: The number of tetrahedra produced with the use of different point selection methods for three models in three dimensions. The first method (CC) always inserts circumcenters of skinny tetrahedra, the second method (OC) always inserts off-centers, and the third method (OPT) is optimization based. The minimal values in each cell are hilighted.

Model	Point position	$\bar{\rho}$									
		2.0	4.0	6.0	8.0	10.0	12.0	14.0	16.0	18.0	20.0
Points in cube	CC	1126	560	527	482	457	429	432	417	417	415
	OC	**777**	**298**	190	184	236	**157**	180	181	**169**	**172**
	OPT	1723	1619	**174**	**174**	**174**	174	**174**	**174**	174	174
Rocket	CC	1017	629	566	**567**	**562**	**542**	**542**	**542**	**542**	**542**
	OC	1060	729	**540**	673	679	660	660	660	660	660
	OPT	**937**	**610**	628	678	598	**542**	**542**	**542**	**542**	**542**
Bat	CC	**24552**	**16561**	**15427**	**15226**	**15083**	**14923**	**14921**	**14923**	**14894**	14950
	OC	24985	21019	20781	20820	19764	19247	21058	17816	18301	21941
	OPT	24628	17599	15970	15533	15267	15053	15074	15084	14939	**14941**

of an optimization-based point insertion method which with every insertion of a Steiner point within a Type I selection disk of a skinny tetrahedron minimizes the signed difference between the number of deleted good quality tetrahedra and the number of deleted poor quality tetrahedra. As opposed to the two-dimensional case, we did not restrict the position of the Steiner point to a specific arc, but instead sampled 1000 points uniformly in spherical coordinates of the Type I selection disk and chose the best one. For our experiments, we used the following three geometries. The "Points in cube" model is the unit cube with two additional points close to its center at 10^{-4} distance from each other. The "Rocket" model is the example PLC supplied with the `Tetgen` distribution. The "Bat" model is described further in the dissertation and is shown in Figure 3.33. Table 2.3 summarizes the results of our experiments. From our point of view, these results do not provide a basis for conclusions, and further research is required.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3.3 Proof of Good Grading with Selection Disks of Type II

Definition 2.8 (Selection disk of Type II in 3D) *If τ is a poor quality tetrahedron with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq 2$, then the* Type II selection disk *for the insertion of a Steiner point that would eliminate τ is the open disk with center c and radius $r(1 - \frac{2}{\bar{\rho}})$.*

Remark 2.8 *Note from definitions 2.7 and 2.8 that the radius of the Type II selection disk is always smaller than the radius of the Type I selection disk of the same skinny tetrahedron because $r(1 - \frac{2}{\bar{\rho}}) = r - \frac{\rho}{\bar{\rho}}2l$ and $\rho > \bar{\rho}$.*

Definition 2.9 (δ-graded Delaunay refinement algorithm in 3D) *If for every tetrahedron τ with circumcenter c, circumradius r, shortest edge length l, and circumradius-to-shortest edge length ratio $\rho = r/l > \bar{\rho} \geq 2$ a Delaunay refinement algorithm selects a Steiner point p_i within the Type II selection disk such that $\|p_i - c\| < r(1 - \delta)$, where*

$$\frac{2}{\bar{\rho}} \leq \delta \leq 1,$$

*we say that this Delaunay refinement algorithm is δ-*graded.

Lemma 2.8 *If p_i is a vertex of the mesh produced by a δ-graded Delaunay refinement algorithm then the following inequality holds:*

$$R(p_i) \geq C \cdot R(\hat{p}_i), \tag{2.21}$$

where we distinguish among the following cases:

(i) $C = \delta\bar{\rho}$ if p_i is a Steiner point chosen within the Type II selection disk of a skinny tetrahedron;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(ii) $C = \frac{1}{\sqrt{2}}$ *if* p_i *is a circumcenter of an encroached constrained face;*

(iii) $C = \frac{1}{\sqrt{2}}$ *if* p_i *is a midpoint of an encroached constrained segment.*

If p_i *is an input vertex, then*

$$R(p_i) \geq \text{lfs}(p_i).$$

Proof: We need to present a new proof only for case (i) since the proof for case (ii) is the same as in Lemma 2.7, and the proofs for all other cases are independent of the choice of the point within the selection disk and are given in [76].

Case (i) Without the loss of generality, let $e(p_l p_m)$ be the shortest edge of the skinny tetrahedron $\tau(p_k p_l p_m p_n)$ and $\hat{p}_i = p_l$. Then

$$\begin{array}{rcll} R(p_i) & > & \delta r(\tau(p_k p_l p_m p_n)) & \text{(from Delaunay property and Def. 2.9)} \\ & = & \delta \rho(\tau(p_k p_l p_m p_n)) \|p_l - p_m\| & \\ & > & \delta \bar{\rho} \|p_l - p_m\| & \text{(since the tetrahedron is skinny)} \\ & \geq & \delta \bar{\rho} R(\hat{p}_i); & \end{array}$$

choose $C = \delta \bar{\rho}$. ■

Similar to the two-dimensional case, the density of vertices near p at the time p is inserted into the mesh, is defined in [76] as

$$D(p) = \frac{\text{lfs}(p)}{R(p)}. \tag{2.22}$$

Lemma 2.9 (Extension of Lemma 2.4 to 3D) *If* p *is a vertex of the mesh produced by a* δ*-graded Delaunay refinement algorithm and* C *is the constant specified by Lemma 2.8, then the following inequality holds:*

$$D(p) \leq \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C}. \tag{2.23}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Proof: If p is inside a Type II selection disk of a skinny tetrahedron with circumradius r, then

$$\begin{aligned} \|p-\hat{p}\| &< 2r-\delta r && \text{(from the definition of } \hat{p} \text{ and Def. 2.9)} \\ &= (2-\delta)r \\ &= \tfrac{2-\delta}{\delta}\delta r \\ &< \tfrac{2-\delta}{\delta}R(p) && \text{(from Delaunay property and Def. 2.9).} \end{aligned}$$

If p is an input vertex or lies on an encroached face or segment, then

$$\begin{aligned} \|p-\hat{p}\| &\le R(p) && \text{(by definitions of } \hat{p} \text{ and } R(p)\text{)} \\ &\le \tfrac{2-\delta}{\delta}R(p) && \text{(since from Def. 2.9, } \delta \le 1\text{).} \end{aligned}$$

In all cases,

$$\|p-\hat{p}\| \le \frac{2-\delta}{\delta}R(p). \tag{2.24}$$

Then

$$\begin{aligned} \operatorname{lfs}(p) &\le \operatorname{lfs}(\hat{p}) + \|p-\hat{p}\| && \text{(from Lemma 2.1)} \\ &\le \operatorname{lfs}(\hat{p}) + \tfrac{2-\delta}{\delta}R(p) && \text{(from (2.24))} \\ &= D(\hat{p})\,R(\hat{p}) + \tfrac{2-\delta}{\delta}R(p) && \text{(from (2.22))} \\ &\le D(\hat{p})\,\tfrac{R(p)}{C} + \tfrac{2-\delta}{\delta}R(p) && \text{(from Lemma 2.8).} \end{aligned}$$

The result follows from the division of both sides by $R(p)$. ■

Lemma 2.10 (Extension of Theorem 5 in [76]) *Suppose that $\bar{\rho} > 2$ and the input PLC satisfies the projection condition. Then there exist fixed constants $C_T \ge 1$, $C_F \ge 1$, and $C_S \ge 1$ such that, for any vertex p inserted or rejected by a δ-graded Delaunay refinement algorithm, the following relations hold:*

(i) $D(p) \le C_T$ if p is chosen from the selection disk of a skinny tetrahedron;

(ii) $D(p) \le C_F$ if p is the circumcenter of an encroached constrained face;

(iii) $D(p) \le C_S$ if p is the midpoint of an encroached constrained segment.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Therefore, the insertion radius of every vertex has a lower bound proportional to its local feature size.

Proof: The proof is by induction and is similar to the proof of Lemma 7 in [76]. The base case covers the input vertices, and the inductive step covers the other three types of vertices: free vertices, the circumcenters of constrained faces, and the midpoints of constrained subsegments.

Base case: The lemma is true if p is an input vertex, because in this case, by Lemma 2.8,

$$D(p) = \frac{\text{lfs}(p)}{R(p)} \leq 1.$$

Inductive hypothesis: Assume that the lemma is true for $\hat{p}$, i.e.,

$$D(\hat{p}) \leq \max\{C_T, C_F, C_S\}.$$

Inductive step: There are three cases:

(i) If p is in the Type II selection disk of a skinny tetrahedron, then

$$\begin{array}{rcll} D(p) & \leq & \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C} & \text{(from Lemma 2.9)} \\ & = & \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{\delta\bar{\rho}} & \text{(from Lemma 2.8)} \\ & \leq & \frac{2-\delta}{\delta} + \frac{\max\{C_T,C_F,C_S\}}{\delta\bar{\rho}} & \text{(by the inductive hypothesis).} \end{array}$$

It follows that one can prove that $D(p) \leq C_T$ if C_T is chosen so that

$$\frac{2-\delta}{\delta} + \frac{\max\{C_T, C_F, C_S\}}{\delta\bar{\rho}} \leq C_T. \tag{2.25}$$

(ii) If p is inserted or considered for insertion at the circumcenter of a constrained face then we have two sub-cases:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(ii-a) If $\hat{p}$ is an input vertex or lies on a non-incident facet or segment, then

$$\operatorname{lfs}(p) \leq R(p)$$

and

$$D(p) \leq 1,$$

thus the lemma holds.

(ii-b) If $\hat{p}$ is a rejected encroaching point inside a Type II selection disk of a skinny tetrahedron then

$$\begin{aligned} D(p) &\leq \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{C} && \text{(from Lemma 2.9)} \\ &= \tfrac{2-\delta}{\delta} + \sqrt{2}D(\hat{p}) && \text{(from Lemma 2.8)} \\ &\leq \tfrac{2-\delta}{\delta} + \sqrt{2}C_T && \text{(by the inductive hypothesis).} \end{aligned}$$

Therefore, one can prove that $D(p) \leq C_F$ if C_F is chosen so that

$$\frac{2-\delta}{\delta} + \sqrt{2}C_T \leq C_F. \tag{2.26}$$

(iii) If p is the midpoint of a subsegment s, then we have two sub-cases:

(iii-a) If $\hat{p}$ is an input vertex or lies on a segment or facet not incident to s, then

$$\operatorname{lfs}(p) \leq R(p)$$

and

$$D(p) \leq 1,$$

thus the lemma holds.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(iii-b) If $\hat{p}$ is a rejected point from the Type II selection disk of a skinny tetrahedron or a rejected encroaching circumcenter of a constrained face then

$$\begin{array}{rcll} D(p) & \leq & \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C} & \text{(from Lemma 2.9)} \\ & = & \frac{2-\delta}{\delta} + \sqrt{2}D(\hat{p}) & \text{(from Lemma 2.8)} \\ & \leq & \frac{2-\delta}{\delta} + \sqrt{2}\max\{C_T, C_F\} & \text{(by the inductive hypothesis).} \end{array}$$

Then one can prove that $D(p) \leq C_S$ if C_S is chosen so that

$$\frac{2-\delta}{\delta} + \sqrt{2}\max\{C_T, C_F\} \leq C_S. \tag{2.27}$$

By Definition 2.9, $\delta \geq \frac{2}{\bar{\rho}}$. If $\delta > \frac{2}{\bar{\rho}}$, inequalities (2.25), (2.26), and (2.27) can be simultaneously satisfied by choosing

$$C_T = C_0\frac{C_1 + 1 + \sqrt{2}}{C_1 - 2}, \quad C_F = C_0\frac{(1+\sqrt{2})C_1 + \sqrt{2}}{C_1 - 2}, \quad C_S = C_0\frac{(3+\sqrt{2})C_1}{C_1 - 2},$$

where

$$C_0 = \frac{2-\delta}{\delta} \quad \text{and} \quad C_1 = \delta\bar{\rho}.$$

As δ approaches $\frac{2}{\bar{\rho}}$, the constants approach infinity. ■

Theorem 2.4 (Theorem 6 in [76]) *For any vertex p of the output mesh, the distance to its nearest neighbor is at least $\frac{\mathrm{lfs}(p)}{C_S+1}$.*

The proof of this theorem does not depend on the specific position of the Steiner point inside the selection disk. Hence, the theorem holds in the context of three-dimensional generalized Delaunay refinement.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

Parallel Generalized Delaunay Refinement

In Section 3.1 we introduce the definition of Delaunay-independent points and prove a Delaunay-independence criterion, i.e., a necessary and sufficient condition. These definition and criterion are applicable to both two-dimensional and three-dimensional parallel Delaunay refinement; they lay the foundation for the development of the sufficient conditions presented in the following sections, both two-dimensional (for the uniform and the graded algorithm) and three-dimensional (for the graded algorithm). Then in Section 3.2 we present our two-dimensional parallel uniform Delaunay refinement algorithm. The sufficient condition in Section 3.2 employs a global circumradius upper bound which helps us construct a uniform lattice for the separation of the regions of concurrent refinement. In Sections 3.3 and 3.4 we describe our parallel graded algorithms, for two dimensions and three dimensions, respectively. Both of the graded algorithms use local sufficient conditions of Delaunay-independence which allow us to use a quadtree or an

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

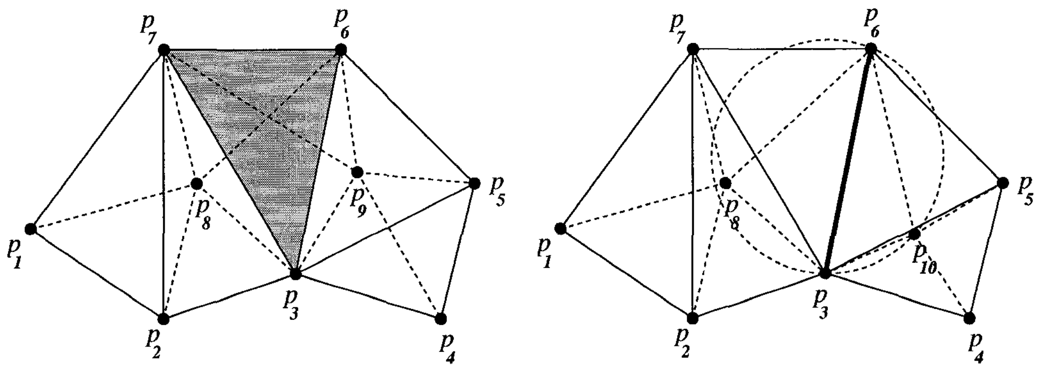


Figure 3.1: Challenges in parallel Delaunay refinement. **(Left)** If $\triangle(p_3p_6p_7) \in \mathcal{C}(p_8) \cap \mathcal{C}(p_9)$, then concurrent insertion of p_8 and p_9 results in a non-conformal mesh. Solid lines represent the edges of the initial triangulation, and dashed lines represent edges created by the insertion of p_8 and p_9. **(Right)** If the edge $e(p_3p_6)$ is shared by $\mathcal{C}(p_8) = \{\triangle(p_1p_2p_7), \triangle(p_2p_3p_7), \triangle(p_3p_6p_7)\}$ and $\mathcal{C}(p_{10}) = \{\triangle(p_3p_5p_6), \triangle(p_3p_4p_5)\}$, then the new triangle $\triangle(p_3p_{10}p_6)$ can have point p_8 inside its circumdisk, thus, violating the Delaunay property.

octree instead of a uniform lattice.

3.1 Delaunay-Independence Criterion

We expect our parallel Delaunay refinement algorithm to insert multiple Steiner points concurrently so that at every iteration the mesh will be both conformal (i.e., simplicial) and Delaunay. Figure 3.1 illustrates how one of these conditions can be violated.

Definition 3.1 (Delaunay-independence) *Points* p_i *and* p_j *are* Delaunay-independent *with respect to mesh* $\mathcal{M}^n = (V^n, T^n)$ *if their concurrent insertion yields the conformal Delaunay mesh* $\mathcal{M}^{n+1} = (V^n \cup \{p_i, p_j\}, T^{n+1})$. *Otherwise,* p_i *and* p_j *are* Delaunay-conflicting.

For practical purposes it is not enough to have the definition of Delaunay-independent points; we need a means to verify whether two points are Delaunay-independent without actually inserting them into the mesh. Below we prove a necessary and sufficient

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

condition (criterion).

Theorem 3.1 (Delaunay-independence criterion) *Points p_i and p_j are Delaunay-independent with respect to mesh $\mathcal{M}^n$ iff both* (3.1) *and* (3.2) *hold:*

$$\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \mathcal{C}_{\mathcal{M}^n}(p_j) = \emptyset, \tag{3.1}$$

$$\forall \xi \in \partial\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}^n}(p_j) \ : \ p_i \notin \bigcirc(p_j \xi). \tag{3.2}$$

Proof: First, $\mathcal{M}^{n+1} = (V^n \cup \{p_i, p_j\}, T^{n+1})$ is conformal iff (3.1) holds. Indeed, if (3.1) holds, then considering (2.2), the concurrent retriangulation of $\mathcal{C}_{\mathcal{M}^n}(p_i)$ and $\mathcal{C}_{\mathcal{M}^n}(p_j)$ will not yield overlapping triangles, and the mesh will be conformal. Conversely, if (3.1) does not hold, the newly created elements will intersect as shown in Figure 3.1(left), and $\mathcal{M}^{n+1}$ will not be conformal.

Now, we will show that $\mathcal{M}^{n+1}$ is Delaunay iff (3.2) holds. The Delaunay Lemma [39] states that iff the empty circumdisk criterion holds for every pair of adjacent elements, then the triangulation is globally Delaunay. Disregarding the symmetric cases, there are three types of pairs of adjacent elements t_r and t_s, where $t_r \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_i)$, that will be affected:

(i) $t_s \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_i)$,

(ii) $t_s \in T^{n+1} \setminus \mathcal{B}_{\mathcal{M}^{n+1}}(p_i) \setminus \mathcal{B}_{\mathcal{M}^{n+1}}(p_j)$, and

(iii) $t_s \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_j)$.

The sequential Delaunay refinement algorithm guarantees that t_r and t_s will be locally Delaunay in the first two cases. In addition, condition (3.2) ensures that they will

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

be locally Delaunay in the third case. Therefore, the mesh will be globally Delaunay. Conversely, if (3.2) does not hold, the elements $(p_i\xi)$ and $(p_j\xi)$ will not be locally Delaunay, and the mesh will not be globally Delaunay. ■

Corollary 3.1 (Sufficient condition of Delaunay-independence) *From Theorem 3.1 it follows that if* (3.1) *holds and*

$$\partial\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}^n}(p_j) = \emptyset, \tag{3.3}$$

then p_i and p_j are Delaunay-independent.

In two dimensions, if p_i encroaches upon a constrained edge, let p_i' be the midpoint of this edge. In three dimensions, if p_i encroaches upon a constrained face, let p_i' the circumcenter of this face, and if p_i or p_i' encroach upon a constrained segment, let p_i'' be the midpoint of this segment (similarly for p_j).

Definition 3.2 (Strong Delaunay-independence) *Points p_i and p_j are* strongly Delaunay-independent *with respect to mesh $\mathcal{M}^n$ iff any pair of points in $\{p_i, p_i', p_i''\} \times \{p_j, p_j', p_j''\}$ are Delaunay-independent with respect to $\mathcal{M}^n$.*

The conditions for strong Delaunay-independence have to be proven separately for two and three dimensions and will appear later in the dissertation in the corresponding chapters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

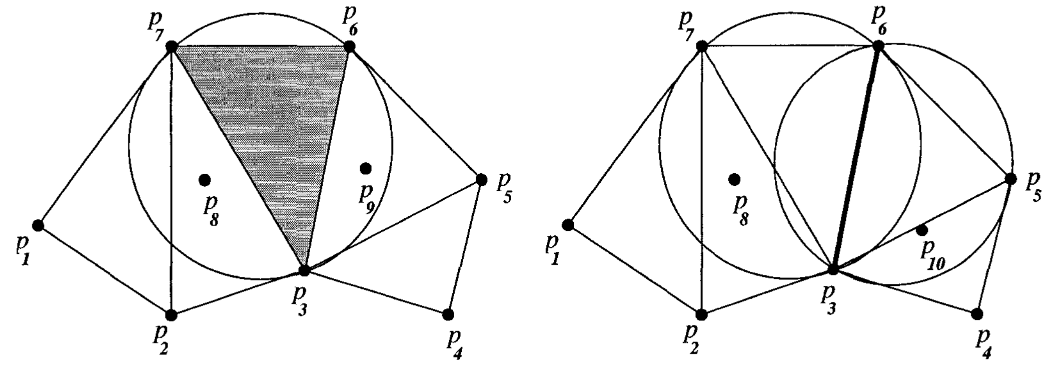


Figure 3.2: Intersecting and adjacent cavities of Steiner points. **(Left)** If $\triangle(p_3p_6p_7) \in \mathcal{C}(p_8) \cap \mathcal{C}(p_9)$, then $\|p_8 - p_9\| < 2r(\triangle(p_3p_6p_7)) < 2\bar{r}$. **(Right)** If edge $e(p_3p_6)$ is shared by the boundaries of $\mathcal{C}(p_8) = \{\triangle(p_1p_2p_7), \triangle(p_2p_3p_7), \triangle(p_3p_6p_7)\}$ and $\mathcal{C}(p_{10}) = \{\triangle(p_3p_5p_6), \triangle(p_3p_4p_5)\}$, then $\|p_8 - p_{10}\| < 2r(\triangle(p_3p_6p_7)) + 2r(\triangle(p_3p_5p_6)) < 4\bar{r}$.

3.2 Two-Dimensional Parallel Uniform Delaunay Refinement

3.2.1 Sufficient Condition of Delaunay-Independence

Theorem 3.1 and Corollary 3.1 are useful in proving other results, but not very efficient in practice, since they require to construct cavities for all candidate points. Previous works [27, 66, 68] employ similar criteria for testing whether a selected subset of Steiner points can be inserted concurrently, and whenever conflicts are found, some of the points are discarded. However, the complexity of this point selection strategy remains to be analyzed and the experience from [27, 66] suggests that the sequential meshing codes cannot be reused due to the necessity to handle setbacks. Below we derive a more practical sufficient condition, which is based on the distance between the points and the circumradii of the triangles. By ensuring that the candidate points are Delaunay-independent, we can avoid setbacks and leverage the existing codes.

Theorem 3.2 (Sufficient Condition of Delaunay-independence) *Let* $\bar{r}$ *be the*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

upper bound on triangle circumradius in the mesh and $p_i, p_j \in \Omega$. *If*

$$\|p_i - p_j\| \geq 4\bar{r}, \tag{3.4}$$

then p_i *and* p_j *are Delaunay-independent.*

Proof: We will prove this theorem by contradiction. Suppose (3.4) holds while p_i and p_j are Delaunay-conflicting. Then from Corollary 3.1, at least one of the following two cases must take place:

(i) Suppose (3.1) does not hold. Then let $\triangle(p_k p_l p_m) \in \mathcal{C}(p_i) \cap \mathcal{C}(p_j)$ (see Figure 3.2(left)). By the definition of cavity,

$$p_i \in \bigcirc(\triangle(p_k p_l p_m)) \quad \text{and} \quad p_j \in \bigcirc(\triangle(p_k p_l p_m)),$$

hence

$$\|p_i - p_j\| < 2r(\triangle(p_k p_l p_m)) < 2\bar{r} < 4\bar{r},$$

which contradicts the assumption that (3.4) holds.

(ii) Suppose (3.3) does not hold. Then let $e(p_k p_l) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$ such that $\triangle(p_k p_l p_m) \in \mathcal{C}(p_i)$ and $\triangle(p_k p_l p_n) \in \mathcal{C}(p_j)$ (see Figure 3.2(right)). Obviously,

$$\|p_k - p_i\| < 2r(\triangle(p_k p_l p_m)) \quad \text{and} \quad \|p_k - p_j\| < 2r(\triangle(p_k p_l p_n)).$$

From the triangle inequality it follows that

$$\|p_i - p_j\| < 2r(\triangle(p_k p_l p_m)) + 2r(\triangle(p_k p_l p_n)) < 4\bar{r},$$

which again contradicts the assumption that (3.4) holds.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

■

Remark 3.1 *The bound* $4\bar{r}$ *can be somewhat decreased by further analysis based on the way* $\bigcirc(\triangle(p_k p_l p_m))$ *and* $\bigcirc(\triangle(p_k p_l p_n))$ *overlap in case (ii). However, the practical value of the additional analysis is limited.*

Theorem 3.2 can be used for an efficient constant-time verification whether two points are independent, and, hence, can be inserted concurrently.

3.2.2 Circumradius Upper Bound Loop Invariant

For Theorem 3.2 to be applicable throughout the run of the algorithm, we need to prove the following invariant:

Loop Invariant 3.1 (Circumradius Upper Bound) *The condition that* $\bar{r}$ *is the upper bound on triangle circumradius in the entire mesh holds both before and after the insertion of a point.*

Next, we show that the execution of the B-W point insertion procedure, either sequentially or in parallel, does not violate Loop Invariant 3.1.

Let the *reflection of disk* $\bigcirc(\triangle(p_k p_l p_m))$ *about the edge* $e(p_k p_l)$ be the disk $\bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$ that has the same radius, whose circle passes through points p_k and p_l, and whose center lies on the other side of edge $e(p_k p_l)$ from point p_m, see Figure 3.3.

Lemma 3.1 *For any point* p_i *inside the region* $\bigcirc(\triangle(p_k p_l p_m)) \setminus \bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$*, see Figure 3.3, the following inequality holds:* $r(\triangle(p_k p_l p_i)) < r(\triangle(p_k p_l p_m))$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

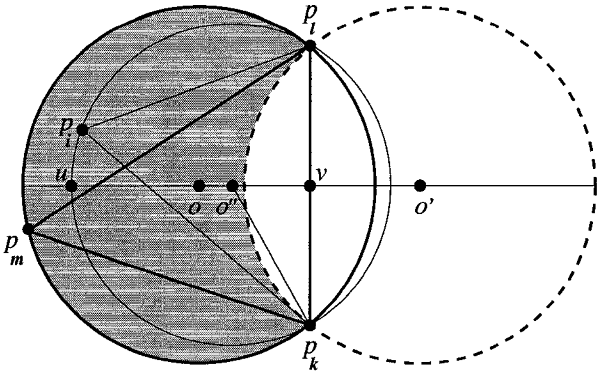


Figure 3.3: The circumradius of a new triangle cannot be larger than the circumradii of the existing triangles. The solid circle corresponds to $\bigcirc(\triangle(p_k p_l p_m))$ with center in point o, and the dashed circle corresponds to $\bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$ with center in point o'. Point o'' is the center of the variable-radius disk $\bigcirc(\triangle(p_i p_k p_l))$, whose circle passes through p_k and p_l. We prove that for any point p_i inside the shaded region, $r(\triangle(p_k p_l p_i)) < r(\triangle(p_k p_l p_m))$.

Proof: Let o'' be the center of the disk $\bigcirc(\triangle(p_k p_l p_i))$, where p_i is any point inside the shaded region $\bigcirc(\triangle(p_k p_l p_m)) \setminus \bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$. o'' has to lie on the straight line through o and o'. Let v be the point of intersection of this straight line with the edge $e(p_k p_l)$ (i.e., the midpoint of $e(p_k p_l)$), and u be the point of intersection of the straight line with the circle of $\bigcirc(\triangle(p_k p_l p_i))$ inside the shaded region. Let $x = \|u - v\|$. We will express the radius of $\bigcirc(\triangle(p_k p_l p_i))$ as a function of x: $r(\triangle(p_k p_l p_i)) = f(x)$.

Let us consider the triangle $\triangle(p_k v o'')$ with the right angle at the vertex v. If we denote $\|p_k - p_l\| = a$, and $\|v - o\| = \|v - o'\| = b$, then by observing that $\|p_k - v\| = a/2$, $\|v - o''\| = x - f(x)$, and $\|o'' - p_k\| = f(x)$, we have:

$$f^2(x) = \left(\frac{a}{2}\right)^2 + (x - f(x))^2, \quad \text{or} \quad f(x) = \frac{a^2}{8x} + \frac{x}{2}, \quad \text{where } 0 < x < \infty.$$

$f(x)$ is convex everywhere on $0 < x < \infty$ since its second derivative is positive:

$$f''(x) = \frac{a^2}{4x^3} > 0 \quad \text{for } 0 < x < \infty.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

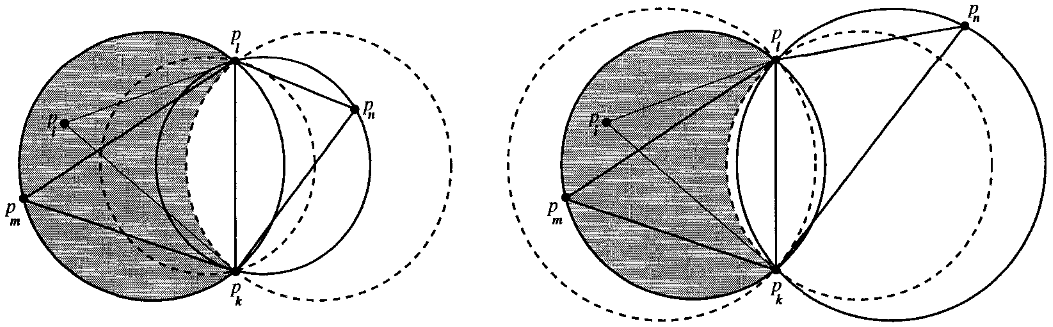


Figure 3.4: The possible relations among the circumradii of the triangles during Delaunay refinement. **(Left)** $\triangle(p_kp_lp_m) \in \mathcal{C}(p_i)$, $\triangle(p_kp_np_l) \notin \mathcal{C}(p_i)$, and $r(\triangle(p_kp_lp_m)) > r(\triangle(p_kp_np_l))$. **(Right)** $\triangle(p_kp_lp_m) \in \mathcal{C}(p_i)$, $\triangle(p_kp_np_l) \notin \mathcal{C}(p_i)$, and $r(\triangle(p_kp_lp_m)) \leq r(\triangle(p_kp_np_l))$.

In addition,

$$\begin{aligned} f(x)|_{o''=o'} &= f(r(\triangle(p_kp_lp_m) - b)) = r(\triangle(p_kp_lp_m)), \\ f(x)|_{o''=o} &= f(r(\triangle(p_kp_lp_m) + b)) = r(\triangle(p_kp_lp_m)). \end{aligned}$$

Using Jensen's inequality, see [91], for $0 \leq \lambda \leq 1$ and

$$r(\triangle(p_kp_lp_m)) - b < x < r(\triangle(p_kp_lp_m)) + b,$$

we can derive the following:

$$\begin{aligned} f(x) &< \lambda f(r(\triangle(p_kp_lp_m)) - b) + (1-\lambda) f(r(\triangle(p_kp_lp_m)) + b) \\ &= \lambda r(\triangle(p_kp_lp_m)) + (1-\lambda) r(\triangle(p_kp_lp_m)) \\ &= r(\triangle(p_kp_lp_m)). \end{aligned}$$

■

Lemma 3.2 *Let* $\triangle(p_kp_lp_m) \in \mathcal{C}(p_i)$ *and* $\triangle(p_kp_np_l) \notin \mathcal{C}(p_i)$. *Then*

$$r(\triangle(p_kp_lp_i)) < \max\{r(\triangle(p_kp_lp_m)), r(\triangle(p_kp_np_l))\}.$$

Proof: There are two cases:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(i) If $r\left(\triangle\left(p_kp_lp_m\right)\right) > r\left(\triangle\left(p_kp_np_l\right)\right)$, see Figure 3.4(left), then p_i has to lie inside the region $\bigcirc\left(\triangle\left(p_kp_lp_m\right)\right) \setminus \bigcirc\left(\triangle\left(p_kp_np_l\right)\right)$, which in this case is a subset of the region $\bigcirc\left(\triangle\left(p_kp_lp_m\right)\right) \setminus \bigcirc'_{e(p_kp_l)}\left(\triangle\left(p_kp_lp_m\right)\right)$, and, according to Lemma 3.1,

$$r\left(\triangle\left(p_kp_lp_i\right)\right) < r\left(\triangle\left(p_kp_lp_m\right)\right).$$

(ii) If $r\left(\triangle\left(p_kp_lp_m\right)\right) \leq r\left(\triangle\left(p_kp_np_l\right)\right)$, see Figure 3.4(right), then p_i has to lie in the region $\bigcirc\left(\triangle\left(p_kp_lp_m\right)\right) \setminus \bigcirc\left(\triangle\left(p_kp_np_l\right)\right)$, which in this case is a subset of $\bigcirc'_{e(p_kp_l)}\left(\triangle\left(p_kp_np_l\right)\right) \setminus \bigcirc\left(\triangle\left(p_kp_np_l\right)\right)$, and, by Lemma 3.1,

$$r\left(\triangle\left(p_kp_lp_i\right)\right) < r\left(\triangle\left(p_kp_np_l\right)\right).$$

■

Theorem 3.3 *Loop Invariant 3.1 holds.*

Proof: The B-W point insertion scheme creates only triangles of the form

$$\triangle\left(p_ip_kp_l\right) \in T^{n+1},$$

where there exists $\triangle\left(p_kp_lp_m\right) \in T^n$ such that $\triangle\left(p_kp_lp_m\right) \in \mathcal{C}\left(p_i\right)$. There are two cases:

(i) There exists $\triangle\left(p_kp_np_l\right) \in \mathcal{M}$ such that $\triangle\left(p_kp_np_l\right) \notin \mathcal{C}\left(p_i\right)$, in other words, $e\left(p_kp_l\right)$ is not a boundary segment. In this case, by Lemma 3.2,

$$r\left(\triangle\left(p_kp_lp_i\right)\right) < \max\{r\left(\triangle\left(p_kp_lp_m\right)\right), r\left(\triangle\left(p_kp_np_l\right)\right)\}.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(ii) $e\left(p_k p_l\right)$ is a boundary segment. Then p_i must be outside of the diametral circle of $e\left(p_k p_l\right)$, otherwise p_i would not have been inserted. Consequently, p_i lies in the region $\bigcirc\left(\triangle\left(p_k p_l p_m\right)\right) \setminus \bigcirc'_{e(p_k p_l)}\left(\triangle\left(p_k p_l p_m\right)\right)$, and by Lemma 3.1,

$$r\left(\triangle\left(p_i p_k p_l\right)\right) < r\left(\triangle\left(p_k p_l p_m\right)\right).$$

■

3.2.3 Adjusting the Degree of Concurrency

We have shown how with a simple and inexpensive test one can check whether two Steiner points can be inserted independently. Now it remains to provide a way of finding enough independent Steiner points, so that all processors can be kept busy inserting them. This task can be accomplished by decreasing all triangle circumradii below some value $\bar{r}$ with the purpose of increasing the number of pairwise independent Steiner points. Fortunately, this can be easily done by applying the sequential Delaunay refinement procedure as a preprocessing step. In this step, we use the initial parameter $\bar{\rho}$ and select parameter $\bar{\mathcal{A}}$ as

$$\bar{\mathcal{A}} = \frac{\bar{r}^2}{4\bar{\rho}^3}, \tag{3.5}$$

where $\bar{r}$ is computed based on the number of available processors and the size of the domain.

Such preprocessing allows us to decrease the maximal circumradius of the triangles in a mesh automatically as a result of decreasing the maximal circumradius-to-shortest edge ratio and the maximal area. In the rest of this subsection, we prove equation (3.5).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We start by proving an auxiliary lemma, which relates the length of any side of a triangle to the sum of the cotangents of adjacent angles and the triangle's area.

Lemma 3.3 *If a, b, c and A, B, C are the lengths of the sides and the measures of the angles of a triangle respectively, then*

$$a^2 = 2(\cot B + \cot C)\mathcal{A}, \tag{3.6}$$

where $\mathcal{A}$ is the area of the triangle.

Proof: The cyclical application of the law of cosines

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc} \tag{3.7}$$

coupled with the formula computing the area of the triangle

$$\mathcal{A} = \frac{1}{2} bc \sin A \tag{3.8}$$

allows us to obtain the following formula:

$$\cot A + \cot B + \cot C = \frac{a^2 + b^2 + c^2}{4\mathcal{A}}. \tag{3.9}$$

Rearranging (3.8) and substituting into (3.7) yields

$$\cot A = \frac{b^2 + c^2 - a^2}{4\mathcal{A}}. \tag{3.10}$$

Finally, by coupling (3.9) and (3.10), we obtain (3.6). ■

Now, we show how the smallest angle of a triangle bounds the sum of the cotangents of the other angles.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Lemma 3.4 *If $A \le B \le C$ are the measures of the angles of a triangle, then*

$$\cot B + \cot C \le \frac{1}{\sin A}.$$

Proof: Since A, B, C are positive and sum up to π, we can denote

$$f(B) = \cot B + \cot C = \cot B + \cot(\pi - A - B) = \cot B - \cot(A + B),$$

where

$$0 < A \le B \le \frac{\pi - A}{2} \le C \le \pi - 2A, \text{ and } A \le \frac{\pi}{3}. \tag{3.11}$$

$f(B)$ is continuous on $\left[A, \frac{\pi - A}{2}\right]$ and has no points of local maxima which satisfy (3.11), thus, it can reach its maxima only in the ends of the interval $\left[A, \frac{\pi - A}{2}\right]$. Direct substitution yields

$$f(A) = \frac{1}{2 \sin A \cos A} \le \frac{1}{2 \sin A \cdot \frac{1}{2}} = \frac{1}{\sin A}$$

and

$$f\left(\frac{\pi - A}{2}\right) = 2\frac{1 - \cos A}{\sin A} \le 2\frac{1 - \frac{1}{2}}{\sin A} = \frac{1}{\sin A}.$$

Hence,

$$f(B) \le \frac{1}{\sin A}, \quad \forall B \in \left[A, \frac{\pi - A}{2}\right].$$

■

Finally, assuming the bounds $\bar{\rho}$ and $\bar{\mathcal{A}}$ on the triangle area and circumradius-to-shortest edge ratio respectively hold over all triangles in a mesh, we find the upper bound $\bar{r}$ on the circumradius of triangles.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Theorem 3.4 *Let a triangle with circumradius r and side lengths $a \leq b \leq c$ have circumradius-to-shortest edge ratio ρ bounded by $\bar{\rho}$: $\rho = r/a < \bar{\rho}$, and area bounded by $\bar{\mathcal{A}}$: $\mathcal{A} < \bar{\mathcal{A}}$. Then*

$$r < 2(\bar{\rho})^{3/2}\sqrt{\bar{\mathcal{A}}}.$$

Proof: From $\rho = r/a$, or $r = \rho a$, using Lemma 3.3 it follows that

$$r = \rho a = \rho\sqrt{2(\cot B + \cot C)\mathcal{A}}$$

and the use of Lemma 3.4 leads to

$$r \leq \rho\sqrt{2\frac{1}{\sin A}\mathcal{A}} = \rho\sqrt{\frac{2\mathcal{A}}{\sin A}}. \tag{3.12}$$

By substituting $\sin A$ from (2.1) into (3.12), we have:

$$r \leq 2\rho^{3/2}\sqrt{\mathcal{A}} < 2(\bar{\rho})^{3/2}\sqrt{\bar{\mathcal{A}}}.$$

■

Theorem 3.5 *If the constants $\bar{\rho}$ and $\bar{\mathcal{A}}$ are upper bounds on triangle area and circumradius-to-shortest edge ratio respectively, then the constant $2(\bar{\rho})^{3/2}\sqrt{\bar{\mathcal{A}}}$ is an upper bound on triangle circumradius, i.e.,*

$$\bar{r} = 2(\bar{\rho})^{3/2}\sqrt{\bar{\mathcal{A}}}.$$

3.2.4 Coarse Grain Partitioning

One possible way to divide the meshing problem of the entire domain up into smaller independent subproblems is to partition the domain into subdomains in such a way that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
ParameterizedDelaunayRefinement($\mathcal{X}$, $\mathcal{M}$, $\bar{\mathcal{A}}$, $\bar{\rho}$, $\sigma()$, $f()$)
Input: $\mathcal{X}$ is a PSLG which defines $\Omega$
       $\mathcal{M}$ is some Delaunay mesh over $\Omega$
       $\bar{\mathcal{A}}$ and $\bar{\rho}$ are desired upper bounds
       $\sigma()$ is a boolean predicate which takes point coordinates as input
       $f()$ is a deterministic function which returns a specific position
           within triangle's selection disk
Output: a conforming Delaunay mesh $\mathcal{M}$ such that
           $\forall t \in \mathcal{M} \ : \ \sigma(f(t)) \Rightarrow (\rho(t) < \bar{\rho}) \wedge (\mathcal{A}(t) < \bar{\mathcal{A}})$
$Q \leftarrow \{t \in \mathcal{M} \mid (\rho(t) \geq \bar{\rho}) \vee (\mathcal{A}(t) \geq \bar{\mathcal{A}})\}$
for each $t \in Q$
    $p_i \leftarrow f(t)$
    if $\sigma(p_i)$
        Find $S$, a set of segments encroached upon by $p$
        if $S \neq \emptyset$
            SplitSegments($S$)
        else
            $\mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{C}(p_i) \cup \{\triangle(p_i p_m p_n) \mid e(p_m p_n) \in \partial\mathcal{C}(p_i)\}$
        endif
        Update $Q$
    endif
endfor
```

Figure 3.5: A sequential Delaunay refinement algorithm with the restriction on the region for the insertion of Steiner points.

the Steiner points of triangles in each subdomain can be inserted concurrently with the Steiner points of triangles in any other subdomain.

If we consider a selected spatial region as a subdomain, which is refined sequentially by one processor, then the layer of the triangles whose Steiner points are within $2\bar{r}$ from any Steiner in the region, forms the buffer zone of the entire region. The partitioning and decoupling of subdomains can be achieved by creating a special buffer zone around every subdomain, so that the insertion of a point in one subdomain can modify the triangles only inside this subdomain and its buffer zone, but the changes will not propagate to other subdomains and their buffer zones. After refining the subdomains, the buffer zones can be refined in a similar way.

We have chosen a simple and efficient way to partition the domain, which consists in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
ParallelDelaunayMeshing($\mathcal{X}$, $\bar{\mathcal{A}}$, $\bar{\rho}$, $P$, $p$, $f()$)
Input: $\mathcal{X}$ is a PSLG which defines $\Omega$
       $\bar{\mathcal{A}}$ and $\bar{\rho}$ are desired upper bounds
       $P$ is the total number of processors (we assume $\sqrt{P}$ is an integer)
       $p$ is the index of the current processor, $0 \le p \le P-1$
       $f()$ is a deterministic function which returns a specific position
           within triangle's selection disk
Output: a distributed conforming Delaunay mesh respecting $\bar{\mathcal{A}}$ and $\bar{\rho}$
Calculate $row(p)$ and $column(p)$ of the current processor
    // $0 \le row(p), column(p) \le \sqrt{P}-1$
Let $l$ be the longest dimension of the bounding box of $\Omega$
$\bar{r} \leftarrow l/(4\sqrt{P})$
if $p = 0$
    Construct $\mathcal{M}_1$, a constrained Delaunay mesh of $\Omega$
    $\bar{\mathcal{A}}_1 \leftarrow \bar{r}^2/(4\bar{\rho}^3)$
    $\mathcal{M}_1 \leftarrow$ ParameterizedDelaunayRefinement($\mathcal{X}$, $\mathcal{M}_1$, $\bar{\mathcal{A}}_1$, $\bar{\rho}$, true, $f()$)
    Assign triangles in $\mathcal{M}_1$ to cells $\{d_{ij} \mid 0 \le i,j \le 4\sqrt{P}-1\}$
        of side length $4\bar{r}$ based on the coordinates of their Steiner points
    for each $i = 0, \ldots, P-1$
        Send cells $\{d_{m,n} \mid 4row(i)-1 \le m \le 4(row(i)+1),$
            $4column(i)-1 \le n \le 4(column(i)+1)\}$ to processor $i$
    endfor
endif
Receive cells $\{d_{ij} \mid 1 \le i,j \le 4\}$ of local mesh $\mathcal{M}$
    // Phase (0) is over
Send cells $\{d_{4,j} \mid 1 \le j \le 3\}$ to processor in $(row(p)+1,\ column(p))$
Send cells $\{d_{i,4} \mid 1 \le i \le 3\}$ to processor in $(row(p),\ column(p)+1)$
Send cell $d_{4,4}$ to processor in $(row(p)+1,\ column(p)+1)$
Receive cells $\{d_{0,j} \mid 1 \le j \le 3\}$ from processor in $(row(p)-1,\ column(p))$
Receive cells $\{d_{i,0} \mid 1 \le i \le 3\}$ from processor in $(row(p),\ column(p)-1)$
Receive cell $d_{0,0}$ from processor in $(row(p)-1,\ column(p)-1)$
    // Phase (1) is over
Let $(x_0, y_0)$ be the upper left coordinate of local cell $d_{0,0}$
$\sigma(q) \leftarrow (q \in [x_0+2\bar{r}, y_0+2\bar{r}] \times [x_0+14\bar{r}, y_0+14\bar{r}])$
$\mathcal{M} \leftarrow$ ParameterizedDelaunayRefinement($\mathcal{X}$, $\mathcal{M}$, $\bar{\mathcal{A}}$, $\bar{\rho}$, $\sigma()$, $f()$)
    // Phase (2) is over
    // Phases (3)–(10) are performed by analogy (Figure 3.9)
return $\mathcal{M}$
```

Figure 3.6: The uniform parallel Delaunay meshing algorithm. In order to simplify the presentation of the algorithm, we do not concern with the index out-of-range problems (e.g., d_{-1}), which we handle in the implementation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

dividing its bounding box up into squares. One of the main advantages of this approach is the efficiency of assigning triangles to subdomains based on the coordinates of the corresponding Steiner points. A small adjustment (see Figure 3.5) to the standard sequential Delaunay refinement algorithm allows the insertion of only those Steiner points that satisfy some user defined condition (i.e., being inside a given rectangle). The drawback is that the amount of work assigned to different processors can vary significantly. However, a runtime system, see e.g. [4], can solve the dynamic load balancing problem and thus we do not have to address it here.

The selection of subdomain box sizes is closely related to the scheduling of refinement phases. First, Theorem 3.2 requires that the regions simultaneously refined by different processors be at least $4\bar{r}$ away. Second, the refinement of a region can create poor quality triangles near the boundary in a neighboring region, which has been refined during a previous iteration. To avoid performing multiple refinement iterations of the same region, some overlapping among the regions needs to be introduced. The degree of overlapping depends on the partitioning of the domain Ω. We partition Ω into a two-dimensional lattice of squares, arrange the processors into logical rows and columns, and assign to each processor a fraction (N/P) of the lattice which it is responsible to mesh, where N is the number of cells and P is the number of processors. We ensure global mesh consistency, by forcing each processor to hold and exchange $(4\bar{r})$-wide layers of triangles near the boundaries of their assigned subdomains. Each processor in position (i, j) communicates only with its neighbors in positions (s, t), such that $|i - s| \leq 1$ and $|j - t| \leq 1$.

The entire parallel algorithm is summarized in Figure 3.6. The scheduling of re-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

finement and communication phases is also schematically presented in Figure 3.9. A sequence of meshes created by four processors for the pipe cross-section model (Figure 3.7) is shown in Figure 3.8.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.2.5 Implementation and Evaluation

For the experimental evaluation of our method, we triangulated (i) 1000 random points uniformly distributed in a unit square and (ii) the pipe cross-section model (Figure 3.7). We used `Triangle` [73] as a sequential mesher on each processor. `Triangle` facilitates the use of a user-defined function for deciding which triangles should be considered "big" and queued for refinement. However, apparently, it does not provide a mechanism for the user to decide at run-time which triangles are "bad" in terms of the circumradius-to-shortest edge ratio or the minimal angle. This fact caused us to insert a test for the location of the given Steiner point inside `Triangle` code and recompile it. We believe that in the future releases of sequential mesh generation codes the facility for defining user-supplied functions, which test the suitability of both types of triangles, can be easily provided.

Another technical detail that we ran into while running our tests, is the substantial difference in time required by `Triangle` to refine an existing triangulation and to construct a fine triangulation from scratch. Contrary to our expectations, building a fine triangulation anew turned out to be much faster. For example, in the case of the contour of the letter "A" which is supplied with `Triangle`, it is about six times faster in retriangulating a given set of points than in reconstructing the mesh data structure from a list of triangles. That is why, when moving a part of the mesh from one processor to another, we decided to migrate only the set of points and boundary edges and retriangulate them as necessary. Since the Delaunay triangulation of a given set of points is unique provided that there are no four cocircular points, this process will not destroy the boundary conformity. In the presence of cocircular points near the boundary,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

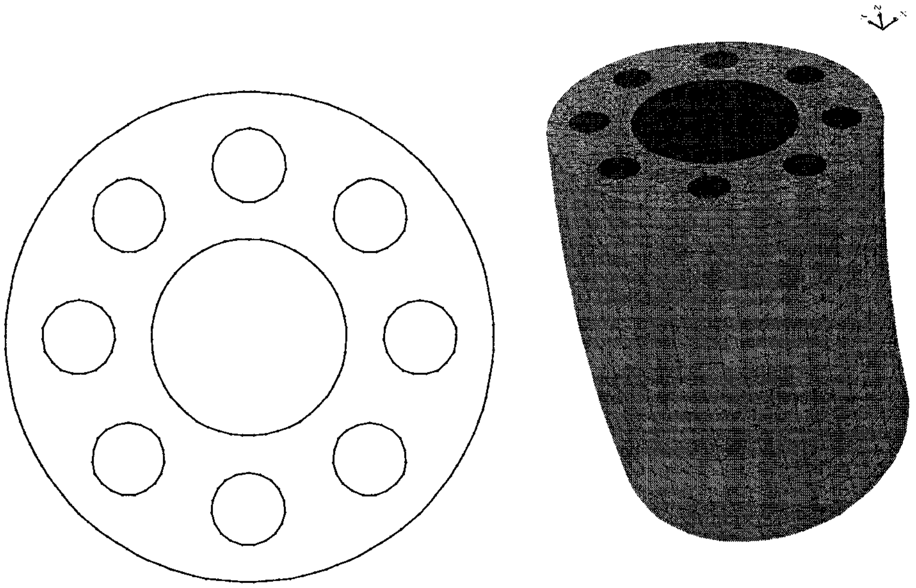


Figure 3.7: A cross-section and a cartoon pipe of a regenerative cooled pipe geometry which came from the testing of a rocket engine at NASA Marshall. NASA had to slightly bend one of the regenerative cooled "pipes" that contained combustion gases in order to fit a rocket engine onto a test stand. In testing, this curved pipe failed due to an unanticipated increase of heating. Since this was a one-off design, NASA could not afford to do extensive testing. The failure caused them to scrap that particular project since it destroyed the engine under test. This is an example of why simulation is useful in reducing cost.

the enforcement of boundary segments causes the required edge flips. The additional advantage of this optimization is the decrease of the network traffic.

We have conducted our experiments on the SciClone cluster computing system at the College of William and Mary using its two tightly-coupled subclusters: "whirlwind" — 64 single-cpu Sun Fire V120 servers (650 MHz, 1 GB RAM) and "twister" — 32 dual-cpu Sun Fire 280R servers (900 MHz, 2 GB RAM). The scaled and the fixed workload evaluation for a mesh of a unit square is shown in Tables 3.1 and 3.2, and for the pipe model — in Table 3.3. We can see that in the pipe case the algorithm performs slightly worse, which happens because of the load imbalance due to the empty space inside pipe holes. Table 3.4 shows a comparison of the distributed memory and the shared memory implementations on the "hurricane" subcluster.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 3.1: Uniform parallel Delaunay refinement of a mesh of a unit square, time measurements for the scaled and the fixed workload. The area bound is inversely proportional to the number of processors (P). For all tests, $\bar{\rho} = 1.41$.

P	Scaled workload			Fixed workload, 23.8M elements, $\bar{A} = 5 \times 10^{-8}$, total time, sec.
	Total Time, sec.	Number of Elements, Millions	Area Bound $\bar{A}$ $\times 10^{-8}$	
4	293.7	23.8	5.00	293.7
9	294.7	58.8	2.22	122.3
16	295.4	109.3	1.25	67.2
25	296.8	175.4	0.80	43.3
36	293.4	255.0	0.56	30.2
49	294.5	352.6	0.41	23.3
64	300.1	470.7	0.31	19.4
81	296.2	587.8	0.25	17.0
100	300.3	738.9	0.20	15.6
121	293.7	873.5	0.17	15.1

Finally, we did not have to write any mesh generation code, since for both the preprocessing and the refinement it was sufficient to call `Triangle` as a library.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 3.2: A breakdown of the execution time for a single processor in the case of the unit square, seconds. The Communication and Synchronization column shows the time to send the data to the neighboring processors and to receive the data from them, which includes waiting for their completion of the current refinement task.

P	Prerefinement with `Triangle`	Total Refinement Time with `Triangle`	Communication and Synchronization	Filling in and Merging Cells
4	0.24	218.7	32.7	39.9
9	0.44	220.0	35.4	40.3
16	0.71	218.8	37.1	40.0
25	1.05	219.1	39.1	40.8
36	1.43	218.6	36.1	40.0
49	1.88	217.7	37.5	40.7
64	2.49	221.0	40.7	41.5
81	3.04	216.6	46.2	41.1
100	3.69	218.4	46.5	41.5
121	4.42	210.3	53.6	40.9

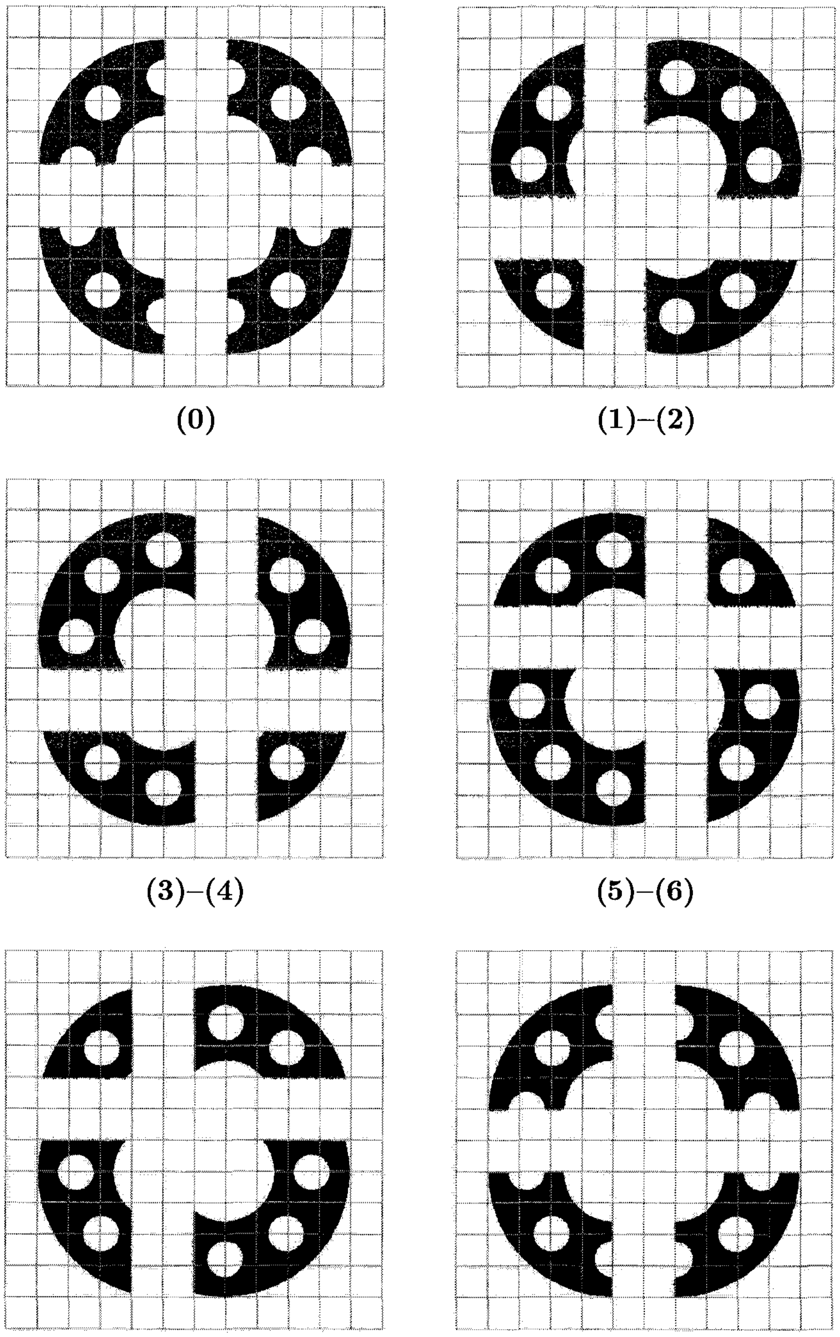
Table 3.3: Uniform parallel Delaunay refinement of a mesh of the pipe model, time measurements for the scaled and the fixed workload. $\bar{\rho} = 1.41$.

P	Scaled workload			Fixed workload, 14.6M elements, $\bar{A} = 2 \times 10^{-2}$, total time, sec.
	Total Time, sec.	Number of Elements, Millions	Area Bound $\bar{A}$, $\times 10^{-2}$	
4	179.1	14.6	2.00	179.1
9	204.8	32.8	0.88	105.6
16	225.7	58.3	0.50	80.5
25	239.8	91.1	0.32	54.2
36	255.3	131.2	0.22	40.1
49	253.7	178.6	0.16	30.2
64	273.6	233.3	0.13	21.9
81	258.6	295.3	0.10	17.1
100	232.2	364.6	0.08	15.0
121	212.7	441.1	0.06	14.8

Table 3.4: Uniform parallel Delaunay refinement of a mesh of the pipe model, time measurements for the distributed (MPI) and the shared (OpenMP) memory implementations.

Number of processors	Time, sec. MPI	Time, sec. OpenMP	Number of Elements, Millions
4	220.3	214.1	14.6

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(0) (1)–(2)

(3)–(4) (5)–(6)

(7)–(8) (9)–(10)

Figure 3.8: Snapshots of a mesh distributed among four processors. Each picture corresponds to the specified phases schematically shown in Figure 3.9. Dark areas are the parts of the mesh that have already been refined. Grey areas are those that have not been refined. Each processor starts with an initial coarse mesh and ends up with a refined mesh covering the same area.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

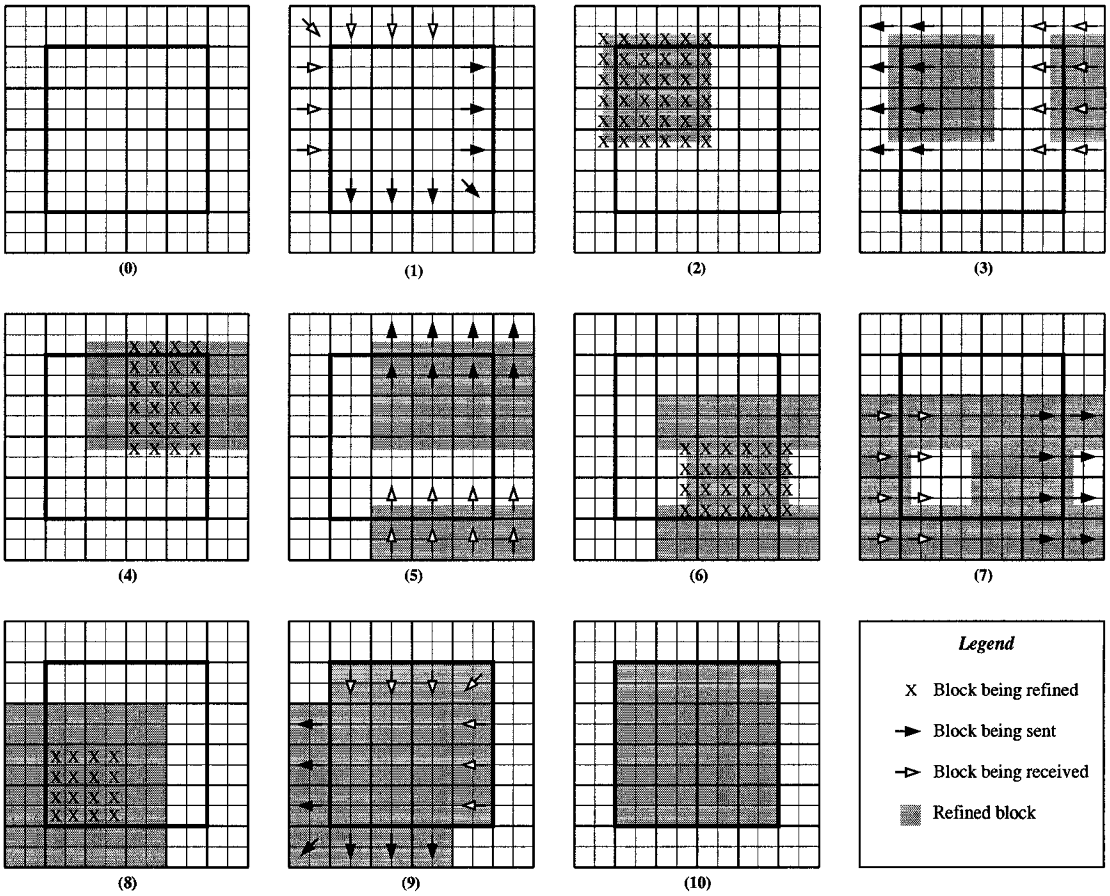


Figure 3.9: A schedule of the uniform parallel Delaunay refinement and communication on distributed memory. Each phase (0)–(10) depicts the actions performed by a single processor. The smallest cells have side length $2\bar{r}$, they are the atomic units of refinement. Since the refinement may leave fractions of big triangles, whose Steiner points are outside the small cell, the refined cells sometimes are not shaded completely. The bigger cells, which consist of 4 small cells, are the atomic units of data migration. The thick lines outline the subdomain assigned to the given processor. A processor also holds parts of its neighbors' meshes. Arrows represent the direction of data migration.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.3 Two-Dimensional Parallel Graded Delaunay Refinement

In this section we extend the uniform parallel Delaunay refinement algorithm for the case when the triangle area bound $\bar{\mathcal{A}}$ is not fixed, but rather changes according to a user-supplied function $\bar{\mathcal{A}}(x,y) : \Omega \to \mathbb{R}$. In particular, in Section 3.3.1 we derive a local sufficient condition of Delaunay independence which does not depend on a global circumradius upper bound. Then in Section 3.3.2 we extend the idea of using a uniform lattice presented in the previous chapter to using a quadtree which reflects the density of the mesh. Based on the proof that the points inserted in specifically chosen separated regions of the quadtree are Delaunay-independent, in Section 3.3.3 we design a parallel algorithm which refines the mesh in multiple regions simultaneously and thus does not require to solve the domain decomposition problem. We describe our implementation of the algorithm and the evaluation of the algorithm's performance. We show that it offers performance improvement over the best available sequential software, even on workstations with just a few hardware cores.

3.3.1 Sufficient Condition of Delaunay-Independence

Lemma 3.5 (Sufficient condition of Delaunay-independence) *Points p_i and p_j are Delaunay-independent if there exists a subsegment s of segment $\mathcal{L}(p_ip_j)$ such that all triangle circumdisks which intersect s have diameter less than or equal to the length of s, i.e.,*

$$\exists s \subseteq \mathcal{L}(p_ip_j) \ : \ \forall t \in T \ : \ s \cap \bigcirc(t) \neq \emptyset \implies 2r(t) \leq |s|, \tag{3.13}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

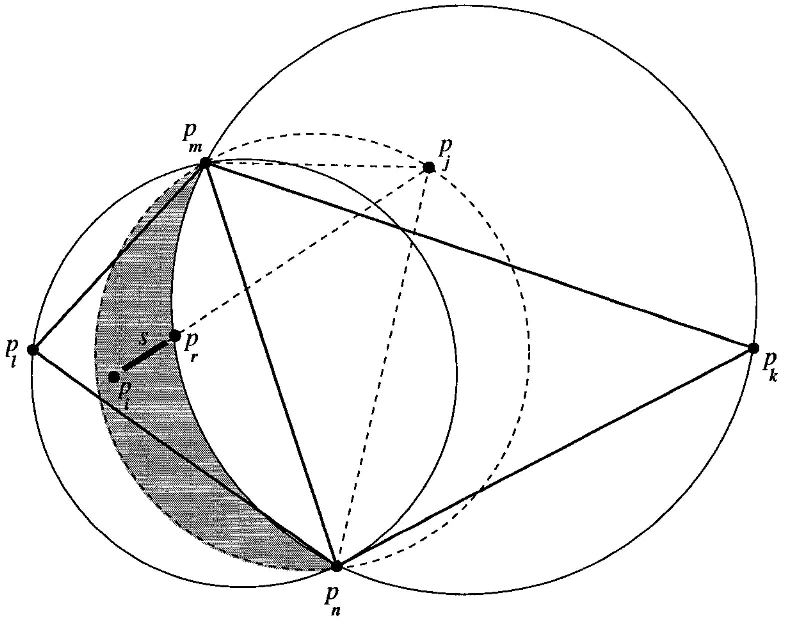


Figure 3.10: An edge shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, first case. $\triangle(p_l p_n p_m) \in \mathcal{C}(p_i)$, $\triangle(p_k p_m p_n) \in \mathcal{C}(p_j)$, $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$, $e(p_m p_n) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, $p_i \in \bigcirc(\triangle(p_j p_m p_n))$, and $r(\triangle(p_k p_m p_n)) > r(\triangle(p_l p_n p_m))$.

where $|s|$ *is the length of* s.

Proof: First, condition (3.13) implies that $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$. Indeed, if there had been a triangle circumdisk which included both p_i and p_j, then the diameter of this circumdisk would be greater than the length of $\mathcal{L}(p_i p_j)$ which would contradict (3.13).

Now, for the boundaries of the cavities there are two possibilities:

(i) If $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) = \emptyset$, then, by Corollary 3.1, p_i and p_j are Delaunay-independent.

(ii) Otherwise, let $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) \neq \emptyset$ and $e(p_m p_n)$ be an arbitrary edge in $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$. We are going to prove that $p_i \notin \bigcirc(\triangle(p_j p_m p_n))$ and, thus, p_i and p_j are Delaunay-independent by Theorem 3.1. The proof is by contradiction. Suppose condition (3.13) holds and $p_i \in \bigcirc(\triangle(p_j p_m p_n))$. There are two cases:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

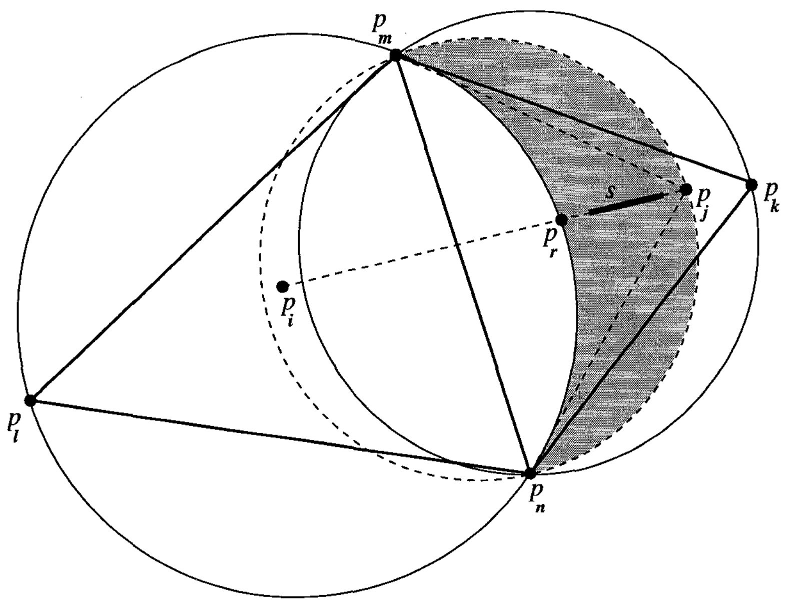


Figure 3.11: An edge shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, second case. $\triangle(p_l p_n p_m) \in \mathcal{C}(p_i)$, $\triangle(p_k p_m p_n) \in \mathcal{C}(p_j)$, $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$, $e(p_m p_n) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, $p_i \in \bigcirc(\triangle(p_j p_m p_n))$, and $r(\triangle(p_k p_m p_n)) \leq r(\triangle(p_l p_n p_m))$.

(ii-a) If $r(\triangle(p_k p_m p_n)) > r(\triangle(p_l p_n p_m))$, see Figure 3.10, then from Lemma 3.2 it follows that

$$r(\triangle(p_j p_m p_n)) < r(\triangle(p_k p_m p_n)). \tag{3.14}$$

In addition, the assumption that $p_i \in \bigcirc(\triangle(p_j p_m p_n))$ implies that

$$|\mathcal{L}(p_i p_j)| < 2r(\triangle(p_j p_m p_n)). \tag{3.15}$$

From (3.14) and (3.15) we conclude that the following relation holds:

$$|\mathcal{L}(p_i p_j)| < 2r(\triangle(p_k p_m p_n)). \tag{3.16}$$

Due to (3.16) and the assumption that (3.13) holds as well as the fact that $|s| \leq |\mathcal{L}(p_i p_j)|$, we conclude that s cannot intersect $\bigcirc(\triangle(p_k p_m p_n))$. If p_r

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is the point of intersection of $\mathcal{L}(p_i p_j)$ with the boundary of $\bigcirc(\triangle(p_k p_m p_n))$, then s is restricted to be a subsegment of $\mathcal{L}(p_i p_r)$ and

$$|s| \leq |\mathcal{L}(p_i p_r)|. \tag{3.17}$$

From the assumptions that $p_i \in \bigcirc(\triangle(p_j p_m p_n))$ and $p_i \notin \bigcirc(\triangle(p_k p_m p_n))$, it follows that p_i has to lie in the crescent-shaped area which is shaded in the figure and the following two relations hold:

$$s \cap \bigcirc(\triangle(p_l p_n p_m)) \neq \emptyset \tag{3.18}$$

and

$$|\mathcal{L}(p_i p_r)| < 2r(\triangle(p_l p_n p_m)). \tag{3.19}$$

Relations (3.17), (3.18), and (3.19) together imply that the condition (3.13) does not hold and we have come to a contradiction.

(ii-b) If $r(\triangle(p_k p_m p_n)) \leq r(\triangle(p_l p_n p_m))$, see Figure 3.11, then from Lemma 3.2 it follows that

$$r(\triangle(p_j p_m p_n)) < r(\triangle(p_l p_n p_m))$$

and considering that

$$|s| \leq |\mathcal{L}(p_i p_j)| < 2r(\triangle(p_j p_m p_n)) < 2r(\triangle(p_l p_n p_m))$$

we conclude that s cannot intersect $\bigcirc(\triangle(p_l p_n p_m))$. This limits s to lie within the subsegment $\mathcal{L}(p_j p_r)$, where p_r is the point of intersection of $\mathcal{L}(p_i p_j)$ with

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the boundary of $\bigcirc\left(\triangle\left(p_l p_n p_m\right)\right)$; therefore,

$$|s| \leq \left|\mathcal{L}\left(p_j p_r\right)\right|. \tag{3.20}$$

The subsegment $\mathcal{L}\left(p_j p_r\right)$ lies completely inside the crescent-shaped region shaded in the figure which in turn is completely inside $\bigcirc\left(\triangle\left(p_k p_m p_n\right)\right)$, hence the following two relations hold:

$$s \cap \bigcirc\left(\triangle\left(p_k p_m p_n\right)\right) \neq \emptyset \tag{3.21}$$

and

$$\left|\mathcal{L}\left(p_j p_r\right)\right| < 2r\left(\triangle\left(p_k p_m p_n\right)\right). \tag{3.22}$$

Relations (3.20), (3.21), and (3.22) together imply that the condition (3.13) does not hold and we have come to a contradiction.

■

3.3.2 Quadtree Construction

Callahan and Kosaraju [9, 10] developed a binary tree data structure for constructing well-separated pair decompositions of points, which was motivated by an application in n-body simulations [40]. They say that point sets A and B are *well-separated* if the rectangles which enclose A and B can each be contained in d-balls of radius r whose minimum distance is at least sr, where s is the *separation.* This data structure is based on a fair split tree of a point set which associates a leaf with each of the points. The construction of the quadtree which we describe below also uses a notion of separated

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

regions. However, in the mesh generation context, the separation is based on the size and the shape of the triangles in the underlying mesh. Another distinction is the introduction of the adjustable *granularity* parameter which allows to reduce the overheads associated with tree updates by increasing the number of triangles per leaf. Finally, unlike in n-body simulations, in mesh refinement we have the creation of new points throughout the execution.

Definition 3.3 (α-neighborhood) *Let the* α-neighborhood $\mathcal{N}_{\alpha}(L_i)$ ($\alpha \in \{\mathit{Left}, \mathit{Right}, \mathit{Top}, \mathit{Bottom}\}$) of quadtree leaf L_i *be the set of all quadtree leaves that share a side with* L_i *and are located in the* α *direction of* L_i.

For example, in Figure 3.12, $L_k \in \mathcal{N}_{Top}(L_i)$ and $L_l \in \mathcal{N}_{Right}(L_i)$.

Definition 3.4 (Orthogonal directions) *Let the* orthogonal directions $\mathrm{ORT}(\alpha)$ *of direction* α *be*

$$\mathrm{ORT}(\alpha) = \begin{cases} \{\mathit{Left}, \mathit{Right}\} & \text{if } \alpha \in \{\mathit{Top}, \mathit{Bottom}\}, \\ \{\mathit{Top}, \mathit{Bottom}\} & \text{if } \alpha \in \{\mathit{Left}, \mathit{Right}\}. \end{cases}$$

Definition 3.5 (Buffer zone) *Let the set of leaves*

$$\mathrm{BUF}(\mathit{Leaf}) = \bigcup_{\alpha} \left(\mathcal{N}_{\alpha}(\mathit{Leaf}) \cup \bigcup_{L \in \mathcal{N}_{\alpha}(\mathit{Leaf})} \; \bigcup_{\beta \in \mathrm{ORT}(\alpha)} \mathcal{N}_{\beta}(L) \right),$$

under the condition that the following relation holds

$$\forall L \in \mathrm{BUF}(\mathit{Leaf}), \forall t \in T \;:\; \bigcirc(t) \cap L \neq \emptyset \implies r(t) < \frac{1}{4}\ell(L), \tag{3.23}$$

where $\ell(L)$ *is the length of the side of* L, *be called a* buffer zone *of leaf Leaf with respect to mesh* $\mathcal{M}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

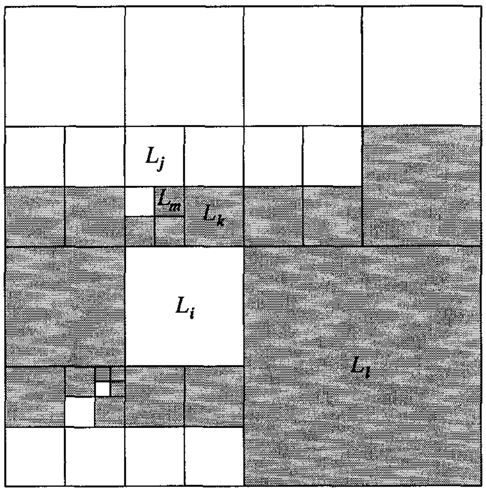


Figure 3.12: An example of the buffer zone BUF (L_i) of a quadtree leaf L_i.

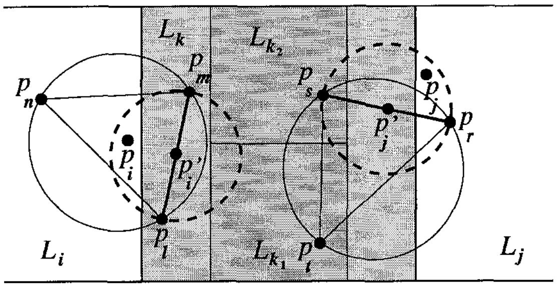


Figure 3.13: Splitting constrained segments and strong Delaunay-independence.

Relation (3.23) is the criterion for the dynamic construction of the quadtree. Starting with the root node which covers the entire domain, each node of the quadtree is split into four smaller nodes as soon as all triangles, whose circumdisks intersect this node, have circumradii smaller than one eighth of its side length.

Definition 3.6 (Delaunay-separated regions) *Let two regions* $R_i \subset \mathbb{R}^2$ *and* $R_j \subset \mathbb{R}^2$ *be called* Delaunay-separated *with respect to mesh* $\mathcal{M}$ *iff any two arbitrary points* $p_i \in R_i$ *and* $p_j \in R_j$ *are strongly Delaunay-independent.*

Lemma 3.6 (Sufficient condition of Delaunay-separateness) *If* L_i *and* L_j *are quadtree leaves,* $i \neq j$*, and* $L_j \notin \mathrm{BUF}\,(L_i)$*, then* L_i *and* L_j *are Delaunay-separated.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

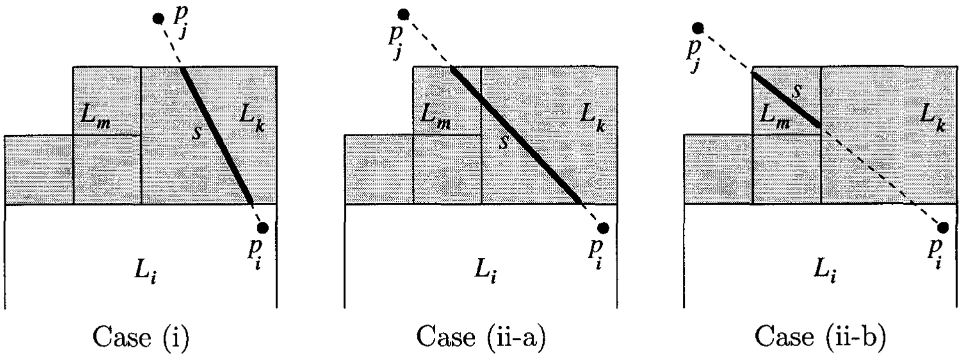


Figure 3.14: All possible positions of Steiner points p_i and p_j, up to symmetry, relative to BUF (L_i).

Proof: First, for an arbitrary pair of points $p_i \in L_i$ and $p_j \in L_j \notin$ BUF (L_i), we will prove that p_i and p_j are Delaunay-independent. Then we will extend the proof to show that any pair of points from $\{p_i, p_i'\} \times \{p_j, p_j'\}$ are Delaunay-independent, which will imply that p_i and p_j are strongly Delaunay-independent; hence, L_i and L_j are Delaunay-separated.

By enumerating all possible configurations of leaves in BUF (L_i) and grouping similar cases, without loss of generality all arrangements can be accounted for using the following argument.

Suppose $\mathcal{L}(p_ip_j)$ intersects the part of the top boundary of L_i that is shared with L_k, where $L_k \in \mathcal{N}_{Top}(L_i) \subset$ BUF (L_i), see Figure 3.14. (The intersection with the other boundaries is symmetric up to rotation.) For each of the following sub-cases we show that there exists a subsegment s of segment $\mathcal{L}(p_ip_j)$ which satisfies the condition of Lemma 3.5, and therefore p_i and p_j are Delaunay-independent:

(i) If $\mathcal{L}(p_ip_j)$ intersects the boundary of L_k which is parallel to the upper boundary of L_i (which can only be the top boundary of L_k), see Figure 3.14(left), then we choose s as the intersection of $\mathcal{L}(p_ip_j)$ with L_k and note that $|s| \geq \ell(L_k)$ by

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

an obvious projection argument. By construction, see Definition 3.5, all triangle circumdisks which intersect L_k have diameter less than $\ell(L_k)$, hence, they also have diameter less than $|s|$.

(ii) Otherwise, let $\mathcal{L}(p_i p_j)$ intersect the boundary of L_k which is orthogonal to the upper boundary of L_i. Suppose this is the left boundary (the intersection with the right boundary is symmetric). Let $L_m \in \mathcal{N}_{Left}(L_k) \subset \mathrm{BUF}(L_i)$ be the leaf adjacent to this boundary at the point of intersection, such that $\ell(L_m) \leq \ell(L_k)$ (otherwise, by symmetry, we switch the roles of L_m and L_k). There are two subcases with respect to the second point of intersection of $\mathcal{L}(p_i p_j)$ with the boundary of L_k:

(ii-a) If $\mathcal{L}(p_i p_j)$ intersects the boundary of L_m which is parallel to the upper boundary of L_i (which can only be the top boundary of L_m), see Figure 3.14 (center), then we select s at the intersection of $\mathcal{L}(p_i p_j)$ with $L_k \cup L_m$. In this case, $|s| \geq \ell(L_k)$ by the assumption that $\ell(L_m) \leq \ell(L_k)$ and by the conformity of the quadtree. By construction, see Definition 3.5, all triangle circumdisks which intersect s have diameter less than $\max\{\ell(L_k), \ell(L_m)\} = \ell(L_k)$, hence, they also have diameter less than $|s|$.

(ii-b) If $\mathcal{L}(p_i p_j)$ intersects the boundary of L_m which is orthogonal to the upper boundary of L_i (it has to be the left boundary, because the first point of intersection is with the right boundary of L_m), see Figure 3.14(right), then we select s at the intersection of $\mathcal{L}(p_i p_j)$ with L_m. In this case, $|s| \geq \ell(L_m)$ and by construction, see Definition 3.5, all triangle circumdisks which intersect s

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

have diameter less than $\ell(L_m)$, hence, they also have diameter less than $|s|$.

Now, suppose p_i and p_j encroach upon constrained edges $e(p_l p_m)$ and $e(p_r p_s)$, respectively, see Figure 3.13. Then the midpoints p'_i and p'_j of $e(p_l p_m)$ and $e(p_r p_s)$ will be inserted instead. If p'_i and p'_j lie in the same quadtree leaves as p_i and p_j respectively, then they can be proven Delaunay-independent using the argument above.

Let us analyze the worst case, i.e., $p'_i, p'_j \in L_k \in \mathrm{BUF}(L_i)$. Since the diametral disk of an edge has the smallest radius among all disks whose circle passes through the endpoints of the edge, then

$$r(e(p_l p_m)) \leq r(\triangle(p_l p_m p_n)) < \frac{1}{4}\ell(L_k)$$

and

$$r(e(p_r p_s)) \leq r(\triangle(p_r p_s p_t)) < \frac{1}{4}\ell(L_k).$$

Therefore, $\|p'_i - p'_j\| > \frac{1}{2}\ell(L_k)$. We can construct imaginary buffer squares L_{k_1} and L_{k_2} between p'_i and p'_j inside leaf L_k with $\ell(L_{k_1}) = \ell(L_{k_2}) = \frac{1}{2}\ell(L_k)$. Since by construction all triangle circumdisks which intersect L_k have radii less than $\frac{1}{4}\ell(L_k)$, then all triangle circumdisks which intersect L_{k_1} or L_{k_2} will have radii less than $\frac{1}{2}\ell(L_{k_1})$ or $\frac{1}{2}\ell(L_{k_2})$. Then, by an argument similar to the one above and since there is no further encroachment, we show that for p'_i and p'_j the condition of Lemma 3.5 is satisfied, and hence these two points are Delaunay-independent. ■

3.3.3 Implementation and Evaluation

If a part of the mesh associated with leaf *Leaf* of the quadtree is scheduled for refinement by a thread, no other thread can refine the parts of the mesh associated with the buffer

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
ParallelGeneralizedDelaunayRefinement($\mathcal{X}$, $P$, $g$, $\bar{\mathcal{A}}()$, $\bar{\rho}$, $f()$)
Input: $\mathcal{X}$ is a PSLG which defines $\Omega$
       $P$ is the maximum number of compute threads
       $g$ is the granularity
       $\bar{\mathcal{A}}()$ is the triangle area grading function
       $\bar{\rho}$ is the upper bound on triangle circumradius-to-shortest edge ratio
       $f()$ is a deterministic function which returns a specific position
           within triangle's selection disk
Output: a conforming Delaunay mesh $\mathcal{M}$ respecting $\bar{\mathcal{A}}()$ and $\bar{\rho}$
Let Quadtree be a quadtree, initially consisting of the root node
    which encloses the entire model
Construct $\mathcal{M}$, a constrained Delaunay triangulation of $\mathcal{X}$
Let $RefinementQ = \{Leaf \in Quadtree \mid PoorTriangles(Leaf) \neq \emptyset\}$
Let $p = 0$ be the number of spawned threads
while $RefinementQ \neq \emptyset$ or $p > 0$
    if $RefinementQ = \emptyset$ or $p = P$
        Wait for a thread to finish refining Leaf
        $p \leftarrow p - 1$
        $RefinementQ \leftarrow RefinementQ \cup$
            $\{L \in \mathrm{BUF}^2(Leaf) \mid |PoorTriangles(L)| > 0\}$
    else
        Let Leaf be the leaf on the top of RefinementQ
        $RefinementQ \leftarrow RefinementQ \setminus \mathrm{BUF}^2(Leaf)$
        $p \leftarrow p + 1$
        spawn GeneralizedDelaunayRefinement($\mathcal{X}$, $g$, $\bar{\mathcal{A}}()$, $\bar{\rho}$, $f()$, $\mathcal{M}$, Leaf)
    endif
endwhile
return $\mathcal{M}$
```

Figure 3.15: The Parallel Generalized Delaunay Refinement algorithm executed by the master thread.

zone BUF (*Leaf*) of this leaf. To simplify the presentation, here we rewrite the definition of the buffer zone in the way it is used by the algorithm. For this purpose, we introduce a superscript to the BUF $(\cdot)$ symbol:

$$\mathrm{BUF}^1(Leaf) = \mathrm{BUF}(Leaf),$$

$$\mathrm{BUF}^i(Leaf) = \mathrm{BUF}^{i-1}(Leaf) \cup \bigcup_{L \in \mathrm{BUF}^{i-1}(Leaf)} \mathrm{BUF}(L), \quad i \geq 2.$$

The algorithm is designed for the execution by one master thread which manages

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
GeneralizedDelaunayRefinement($\mathcal{X}$, $g$, $\bar{\mathcal{A}}()$, $\bar{\rho}$, $f()$, $\mathcal{M}$, Leaf)
Input:  See the PGDR algorithm in Figure 3.15
Output: Locally refined Delaunay mesh $\mathcal{M}$
        Locally refined quadtree node Leaf
 1  $i_{min} \leftarrow \min_{PoorTriangles_i(Leaf) \neq \emptyset} i$
 2  $i_{max} \leftarrow i_{min} + g$
 3  for $j = i_{min}, \dots, i_{max}$
 4      while $PoorTriangles_j(Leaf) \neq \emptyset$
 5          Let $t \in PoorTriangles_j(Leaf)$
 6          $p \leftarrow f(t)$
 7          Insert $p$ into $\mathcal{M}$
 8          for $L \in \{Leaf\} \cup \mathrm{BUF}(Leaf)$
 9              Update $PoorTriangles(L)$ and $Counter(L)$
10          endfor
11      endwhile
12  endfor
13  Split Leaf recursively while (3.23) holds
14  return $\mathcal{M}$, Leaf
```

Figure 3.16: The Parallel Generalized Delaunay Refinement algorithm executed by each of the refinement threads.

the work pool and multiple refinement threads which refine the mesh and the quadtree. Figure 3.15 presents a high level description of the Parallel Generalized Delaunay Refinement (PGDR) algorithm performed by the master thread. Line 14 shows the invocation of a refinement thread from the master thread. Figure 3.16 presents the part of the algorithm executed by each of the refinement threads.

When a quadtree leaf *Leaf* is scheduled for refinement, we remove not just BUF^1 (*Leaf*) but BUF^2 (*Leaf*) from the refinement queue. Although this is not required by our theory, there are two implementation considerations for doing so, and both are related to the goal of reducing fine-grain synchronization.[1] First, each leaf has an associated data structure which stores the poor quality triangles whose circumdisks intersect this leaf, so that we can maintain the relation (3.23). Therefore, we would have to introduce synchronization in line 9 of the algorithm in Figure 3.16 to maintain

---

[1] As we have shown previously [3], the overheads of portable thread packages (e.g., Pthreads) on modern SMTs are not small enough to tolerate fine-grain parallelism in Delaunay mesh refinement.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

this data structure. Second, for efficiency considerations, we followed the design of the triangle data structure that is used in `Triangle` [73]. In particular, each triangle contains pointers to neighboring triangles for easy mesh traversal. However, if two cavities share an edge and are updated by the concurrent threads, which in theory can be done legitimately in certain cases, these triangle–neighbor pointers will be invalidated. For these reasons, we chose to completely separate the sets of leaves affected by the mesh refinement performed by multiple threads.

Each of the worker threads performs the refinement of the mesh and the refinement of the quadtree. The poor quality triangles whose split-points selected by a deterministic function $f()$ are inside the square of *Leaf* are stored in the data structure denoted here as *PoorTriangles*(*Leaf*). *Leaf* needs to be scheduled for refinement if this data structure is not empty. In addition, each *Leaf* has a counter for the triangles with various ratios of the side length of *Leaf* to their circumradius. If we denote $\phi(t, Leaf) = \left\lfloor \log_2 \frac{\ell(Leaf)}{r(t)} \right\rfloor$, then $Counter_i(Leaf) = |\{t \in \mathcal{M} \mid (\bigcirc(t) \cap Leaf \neq \emptyset) \wedge (\phi(t, Leaf) = i)\}|$. When $Counter_i(Leaf) = 0$, $\forall i = 1, 2, 3$, it implies that (3.23) would hold for each of the children of *Leaf*, and *Leaf* can be split. Lemma 3.2 guarantees that when a point is inserted into a Delaunay mesh using the B-W algorithm, the circumradii of the new triangles are not going to be larger than the circumradii of the triangles in the cavity of the point or those that are adjacent to the cavity. In addition, the following lemma proves that the circumdisks of the new triangles are not going to extend beyond the circumdisks of the triangles in the cavity and the circumdisks of the triangles adjacent to the cavity. Therefore, newly created triangles will not violate (3.23).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

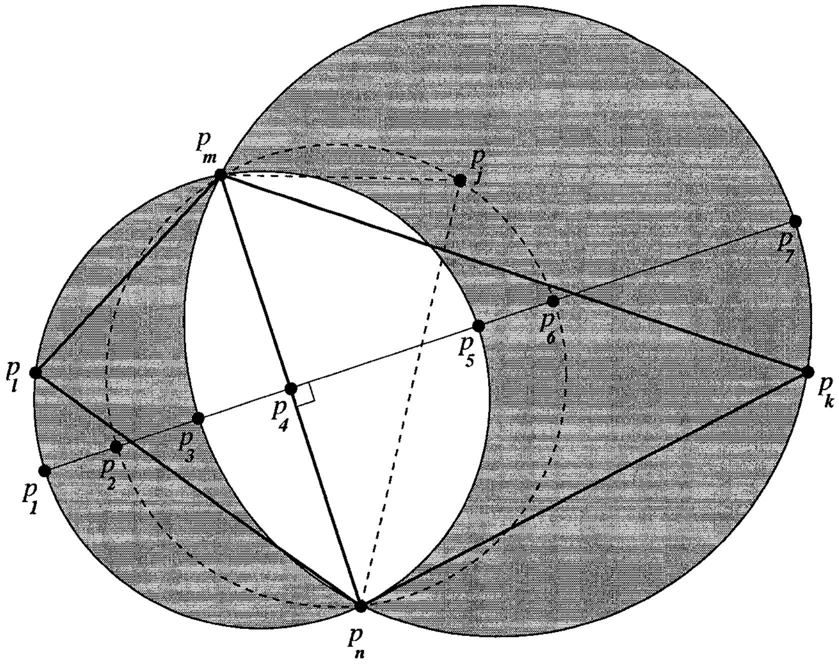


Figure 3.17: The circumdisk of a new triangle is a subset of the union of the circumdisks of the existing triangles.

Lemma 3.7 *Let* $\triangle(p_k p_m p_n) \in \mathcal{C}(p_j)$ *and* $\triangle(p_l p_n p_m) \notin \mathcal{C}(p_j)$. *Then*

$$\bigcirc(\triangle(p_j p_m p_n)) \subset \left(\bigcirc(\triangle(p_k p_m p_n)) \cup \bigcirc(\triangle(p_l p_n p_m)) \right).$$

Proof: Consider Figure 3.17. Let p_4 be the midpoint of edge $e(p_m p_n)$, and let the following points lie at the intersections of the perpendicular bisector of $e(p_m p_n)$ with the boundaries of the corresponding circumdisks: p_1 and p_5 with $\bigcirc(\triangle(p_l p_n p_m))$, p_2 and p_6 with $\bigcirc(\triangle(p_j p_m p_n))$, and p_3 and p_7 with $\bigcirc(\triangle(p_k p_m p_n))$. Due to the premise that $p_j \in \bigcirc(\triangle(p_k p_m p_n))$ and $p_j \notin \bigcirc(\triangle(p_l p_n p_m))$, p_6 is restricted to lie between p_5 and p_7 by construction, and p_2 is restricted to lie between p_1 and p_3 by Delaunay property. Therefore, the arc $p_n p_6 p_j p_m$ is restricted to lie within the shaded region which is $\bigcirc(\triangle(p_k p_m p_n)) \setminus \bigcirc(\triangle(p_l p_n p_m))$ and the arc $p_m p_2 p_n$ is restricted to lie within

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the shaded region which is $\bigcirc\left(\triangle\left(p_l p_n p_m\right)\right) \setminus \bigcirc\left(\triangle\left(p_k p_m p_n\right)\right)$. Hence, $\bigcirc\left(\triangle\left(p_j p_m p_n\right)\right)$ cannot extend beyond $\bigcirc\left(\triangle\left(p_k p_m p_n\right)\right) \cup \bigcirc\left(\triangle\left(p_l p_n p_m\right)\right)$. ■

Each leaf of the quadtree has associated with it a bucketing structure which holds poor quality triangles:

$$\begin{aligned} \mathit{PoorTriangles}_i(\mathit{Leaf}) = \{t \in \mathcal{M} \mid (f(t) \in \mathit{Leaf}) \wedge (\phi(t, \mathit{Leaf}) = i) \wedge \\ ((\mathcal{A}(t) > \bar{\mathcal{A}}(t)) \vee (\rho(t) > \bar{\rho}))\}. \end{aligned}$$

At each mesh refinement step, all triangles in $\mathit{PoorTriangles}_j(\mathit{Leaf})$ are refined, for all $j = i_{min}, \ldots, i_{min} + \mathit{granularity}$, where $i_{min} = \min_{\mathit{PoorTriangles}_i(\mathit{Leaf}) \neq \emptyset} i$, and $\mathit{granularity} \geq 1$ is a parameter that controls how much computation is done during a single mesh refinement call. After a mesh refinement call returns, the feasibility of splitting *Leaf* is evaluated, and it is recursively subdivided if necessary.

The *PoorTriangles* structure allows our parallel algorithm to give priority to triangles with large circumradii. As discussed in [70, 77], Ruppert's sequential Delaunay refinement algorithm has quadratic worst-case running time, even though in most practical cases the time is linear with respect to the output size. Recently, Miller [62] proposed a Delaunay refinement algorithm which runs in optimal $\mathcal{O}(n \log n + m)$ time, where n is the size of the input, and m is the size of the output. He achieved this improvement by introducing a priority queue, where the skinny triangles are ordered by their diameter (equivalently, circumradius), and the triangles with the largest diameter are refined first. Although our algorithm does not introduce total ordering as Miller's sequential algorithm, it allows to eliminate quadratic running time for pathological input geometries.

We developed two implementations of the PGDR algorithm. The first one is written

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

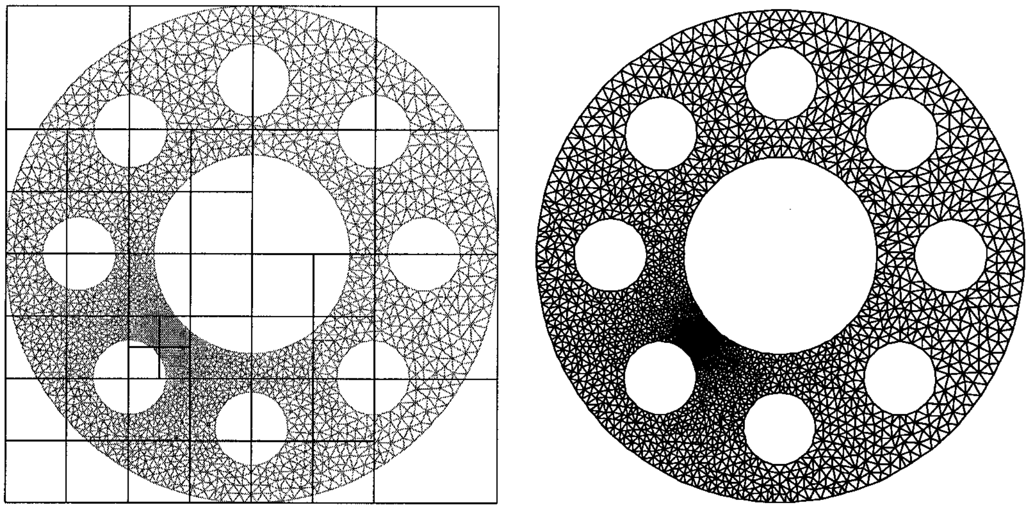


Figure 3.18: The meshes of the pipe cross-section model generated with PGDR and with `Triangle`, $\bar{A}(x,y) = 0.4\sqrt{(x-200)^2+(y-200)^2}+1$. **(Left)** Our parallel refinement algorithm, 4166 triangles. **(Right)** `Triangle` [73], 4126 triangles.

in Python using Python threads module, and the second one is in C++ using Pthreads. The Python code is interpreted and, thus, is much slower than the compiled code written in languages like C/C++, however, it offers high level data types and expressions which allow to significantly decrease the development cycle. We ran this code on a Linux box with two single-core Pentium-4 processors. Figures 3.18 and 3.19 compare the meshes produced by our Python implementation and `Triangle` library [73] for a pipe cross-section and a key. Figure 3.20 also shows the initial geometry and the quadtree produced by our algorithm for the cylinder flow problem [34]. For all of the quadtree nodes, mesh refinement and node subdivision routines were applied concurrently while preserving the required buffer zones, until the quality constraints were met. The specified grading functions are used as follows. If (x_i, y_i) is the centroid of triangle t_i, then the area of t_i has to be less than $\bar{A}(x_i, y_i)$. In all experiments we used the same minimal angle bound of 20°. These tests indicate that while maintaining the required quality of the elements, the number of triangles produced by our method is close, and sometimes is even smaller,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

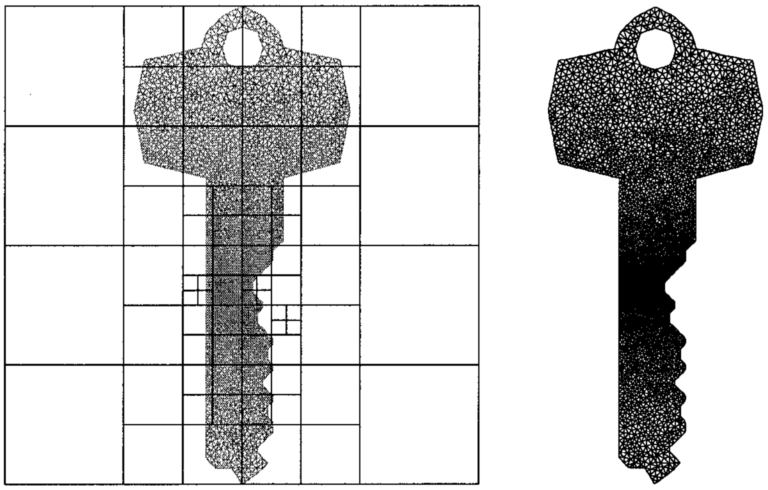


Figure 3.19: The meshes of Jonathan Shewchuk's key model generated with PGDR and with `Triangle`, $\bar{\mathcal{A}}(x,y) = 0.02|y-46|+0.1$. **(Left)** Our parallel refinement algorithm, 5411 triangles. **(Right)** `Triangle` [73], 5723 triangles.

than produced by `Triangle`.

The experiments with the code written in C++ were conducted on an IBM Power5 node with two dual-core processors running at 1.6 GHz and 8 GBytes of total physical memory. We compared our implementation with the fastest to our knowledge sequential Delaunay mesh generator `Triangle` version 1.6 [73]. This is the latest release of `Triangle`, which uses the off-center point insertion algorithm [86]. In order to make the results comparable, our PGDR implementation also uses the off-center point insertion [86]. `Triangle` provides a convenient facility for the generation of meshes respecting user-defined area bounds. The user can write his own `triunsuitable()` function and link it against `Triangle`. This function is called to examine each new triangle and to decide whether or not it should be considered big and enqueued for refinement. We encoded our grading function into the `triunsuitable()` function, compiled it into an object file, and linked against both `Triangle` and our own PGDR implementation. We ran each of the tests 10 times and used average or median timing measurements as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

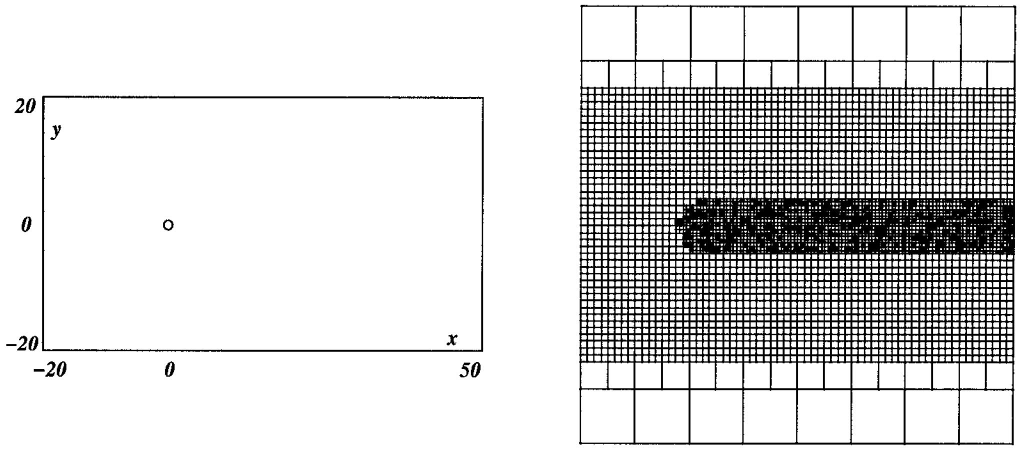


Figure 3.20: The cylinder flow model and its mesh generated with PGDR. $\bar{A}(x,y) = 1.2 \cdot 10^{-3}$ if $((x \geq 0) \wedge (|y| < 5)) \vee ((x < 0) \wedge (\sqrt{x^2 + y^2}) < 5)$; $\bar{A}(x,y) = 10^{-2}$, otherwise. Our parallel refinement algorithm produced 1044756 triangles, and `Triangle` produced 1051324 triangles. **(Left)** The input model. **(Right)** The final quadtree. The complete triangulation is not drawn.

indicated.

Figure 3.21(left) presents the total running times for several granularity values, as the number of compute threads increases from 1 to 4. One additional thread was used to manage the refinement queue. The mesh was constructed for the pipe cross-section model shown in Figure 2.5(left), using the grading function

$$\bar{A}(x,y) = 10^{-4}(\sqrt{(x-200)^2 + (y-200)^2} + 1).$$

The total number of triangles produced both by `Triangle` and by PGDR was approximately 17 million.

We can see that the best running time was achieved using 4 compute threads with the granularity value equal to 2, and it amounted to 56% of `Triangle`'s sequential running time. It is also interesting to see the intersection of lines corresponding to granularities 2 and 3, when the number of compute threads was increased from 3 to 4. This intersection reflects the tradeoff between granularity and concurrency: in order

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

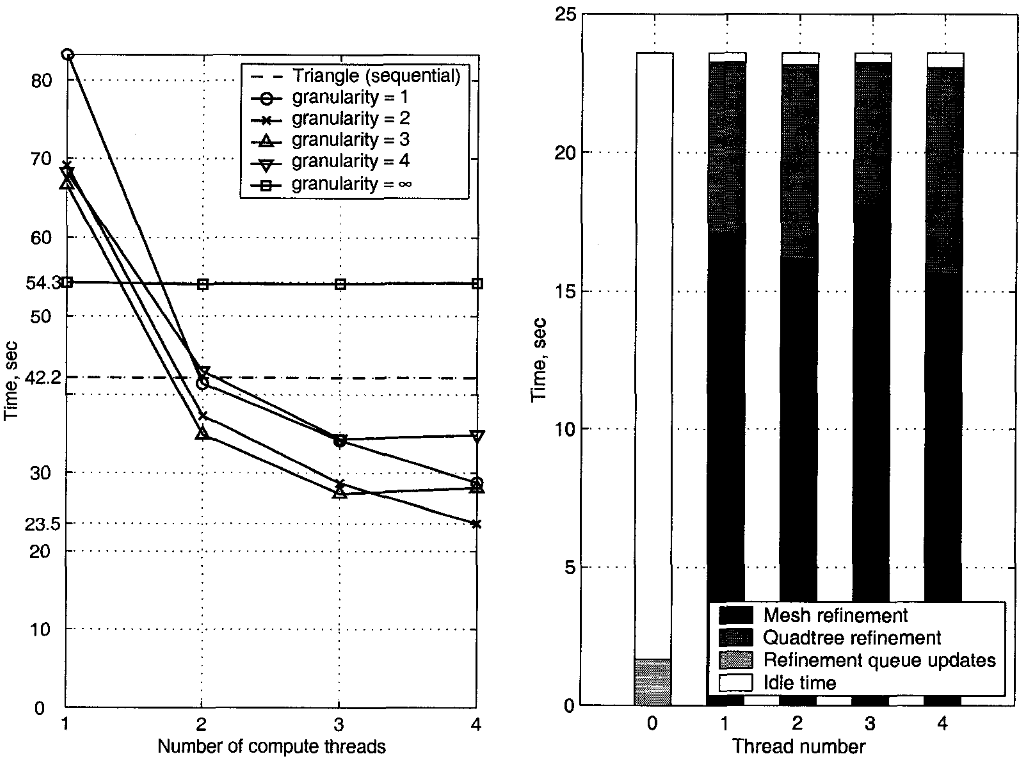


Figure 3.21: **(Left)** The total running time of the PGDR code, for different granularity values, as the number of compute threads is increased form 1 to 4, compared to `Triangle`. Each point on the graph is the average of 10 measurements. **(Right)** The breakdown of the total PGDR execution time for each of the threads, when the number of compute threads is 4 and granularity is 2. Thread number 0 performs only the management of the refinement queue, and threads 1–4 perform mesh and quadtree refinement. The data correspond to the test with the median total running time.

to increase the concurrency we have to decrease the granularity, which introduces more overheads.

Figure 3.21(right) shows the breakdown of the total execution time for each of the threads. The fact that the management thread is idle for 93% of the total time suggests the possibility of high scalability of the code on larger machines, since it can handle many more refinement threads (cores) than the current widely available machines have.

Standard system memory allocators exhibited high latency and poor scalability in our experiments, which lead us to develop a custom memory management class. At

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

initialization, our memory pool class takes the size of the underlying object (triangle, vertex, quadtree node, or quadtree junction point) as a parameter and at runtime it allocates blocks of memory which can fit a large number of objects. When the objects are deleted, they are not deallocated but are kept for later reuse instead. Each thread manages a separate set of memory pools, which allows us to avoid synchronization. Our quantitative study of the performance of the standard, the custom, and a novel generic multiprocessor allocator appears elsewhere [14].

3.4 Three-Dimensional Parallel Graded Delaunay Refinement

3.4.1 Sufficient Condition of Delaunay-Independence

Similar to the two-dimensional case, let the *reflection of disk* $\bigcirc\left(\tau\left(p_m p_k p_l p_n\right)\right)$ *about the face* $\triangle\left(p_k p_l p_n\right)$ be the disk $\bigcirc'_{\triangle(p_k p_l p_n)}\left(\tau\left(p_m p_k p_l p_n\right)\right)$ that has the same radius, whose sphere passes through the circle of $\bigcirc\left(\triangle\left(p_k p_l p_n\right)\right)$, and whose center lies on the other side of $\triangle\left(p_k p_l p_n\right)$ from point p_m.

The following lemma extends Lemma 3.1 to three dimensions.

Lemma 3.8 *For any point* p_i *inside the region*

$$\bigcirc\left(\tau\left(p_m\xi\right)\right) \setminus \bigcirc'_{\xi}\left(\tau\left(p_m\xi\right)\right),$$

where $\xi = \triangle\left(p_k p_l p_n\right)$*, the following inequality holds:*

$$r\left(\tau\left(p_i\xi\right)\right) < r\left(\tau\left(p_m\xi\right)\right).$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

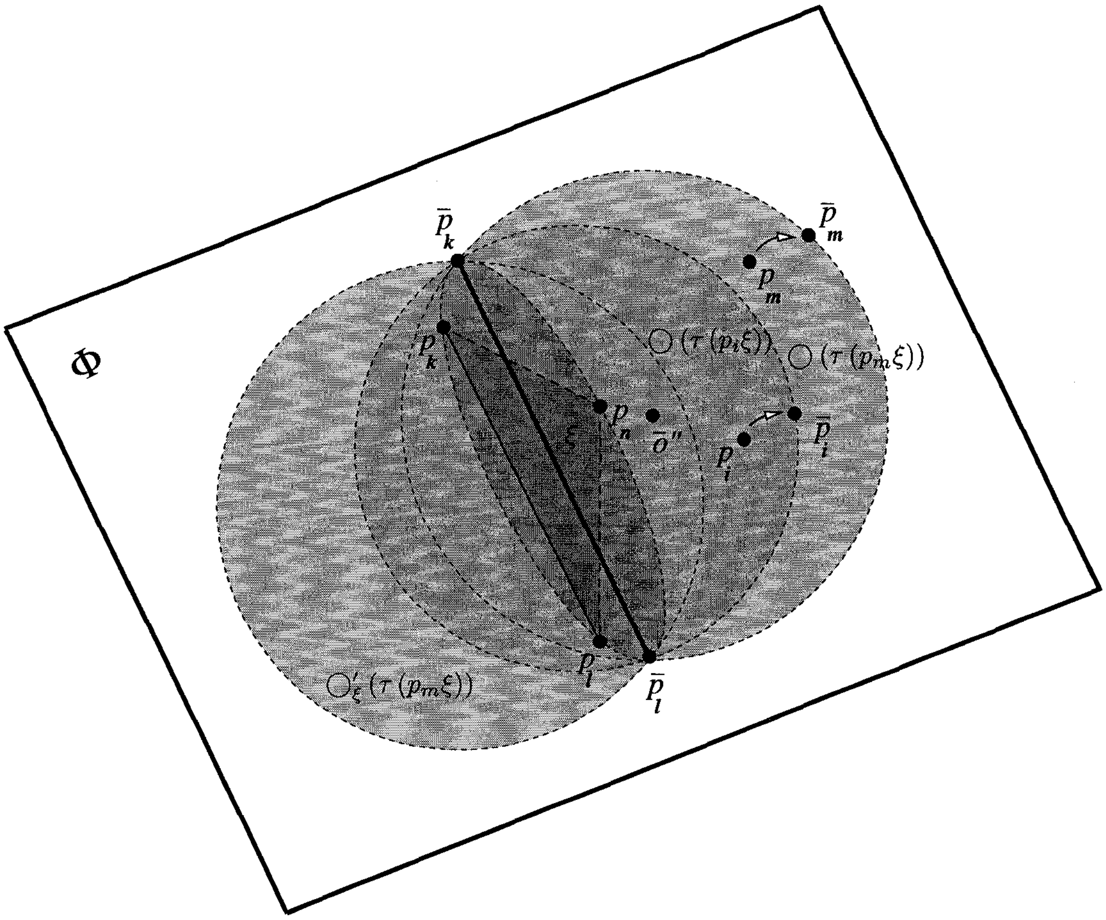


Figure 3.22: A three dimensional disk and its reflection cut by a plane.

Proof: We reduce the three-dimensional case to the two-dimensional case by the following construction (see Figure 3.22). Draw an arbitrary diameter $\mathcal{L}(\bar{p}_k\bar{p}_l)$ of disk $\bigcirc(\xi)$. Then let Φ be the plane which passes through $\mathcal{L}(\bar{p}_k\bar{p}_l)$ and is perpendicular to the plane containing ξ.

Let $\bigcirc(\bar{p}_k\bar{p}_l\bar{p}_i)$ be the intersection of Φ with $\bigcirc(\tau(p_i\xi))$ such that $\bar{p}_i \in \Phi$ is obtained by moving p_i along the surface of the sphere of $\bigcirc(\tau(p_i\xi))$ in the plane perpendicular to Φ.

Let $\bigcirc(\bar{p}_k\bar{p}_l\bar{p}_m)$ be the intersection of Φ with $\bigcirc(\tau(p_m\xi))$ such that $\bar{p}_m \in \Phi$ is obtained by moving p_m along the surface of the sphere of $\bigcirc(\tau(p_i\xi))$ in the plane

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

perpendicular to Φ.

Also, let $\bar{o}$" be the center of the disk $\bigcirc(\tau(p_m\xi))$. Note that $\bar{o}$" $\in \Phi$ because $\mathcal{L}(\bar{p}_k\bar{p}_l)$ is a diameter of $\bigcirc(\xi)$.

By construction, since $\bigcirc(\bar{p}_i\bar{p}_k\bar{p}_l)$ is a two-dimensional equatorial disk of the three-dimensional disk $\bigcirc(\tau(p_i\xi))$, we have:

$$r(\tau(p_i\xi)) = r(\triangle(\bar{p}_i\bar{p}_k\bar{p}_l)), \tag{3.24}$$

and, since $\bigcirc(\bar{p}_m\bar{p}_k\bar{p}_l)$ is a two-dimensional equatorial disk of the three-dimensional disk $\bigcirc(\tau(p_m\xi))$, we have:

$$r(\tau(p_m\xi)) = r(\triangle(\bar{p}_m\bar{p}_k\bar{p}_l)). \tag{3.25}$$

Now we can see that the arrangement on the plane Φ is similar to the two-dimensional arrangement in Figure 3.3 with each point p in two dimensions corresponding to point $\bar{p}$ in the plane Φ. According to the two-dimensional result of Lemma 3.1,

$$r(\triangle(\bar{p}_i\bar{p}_k\bar{p}_l)) < r(\triangle(\bar{p}_m\bar{p}_k\bar{p}_l)). \tag{3.26}$$

Combining relation (3.26) with relations (3.24) and (3.25), we conclude the proof. ■

The following lemma extends Lemma 3.2 to three dimensions:

Lemma 3.9 *Let* $\tau(p_m\xi) \in \mathcal{C}(p_i)$ *and* $\tau(p_r\xi) \notin \mathcal{C}(p_i)$*, where* $\xi = \triangle(p_kp_lp_n) \in \partial\mathcal{C}(p_i)$*. Then*

$$r(\tau(p_i\xi)) < \max\{r(\tau(p_m\xi)), r(\tau(p_r\xi))\}.$$

Proof: The conditions $\tau(p_m\xi) \in \mathcal{C}(p_i)$ and $\tau(p_r\xi) \notin \mathcal{C}(p_i)$ imply that p_i lies inside the region $\Psi = \bigcirc(\tau(p_m\xi)) \setminus \bigcirc(\tau(p_r\xi))$. There are two cases:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

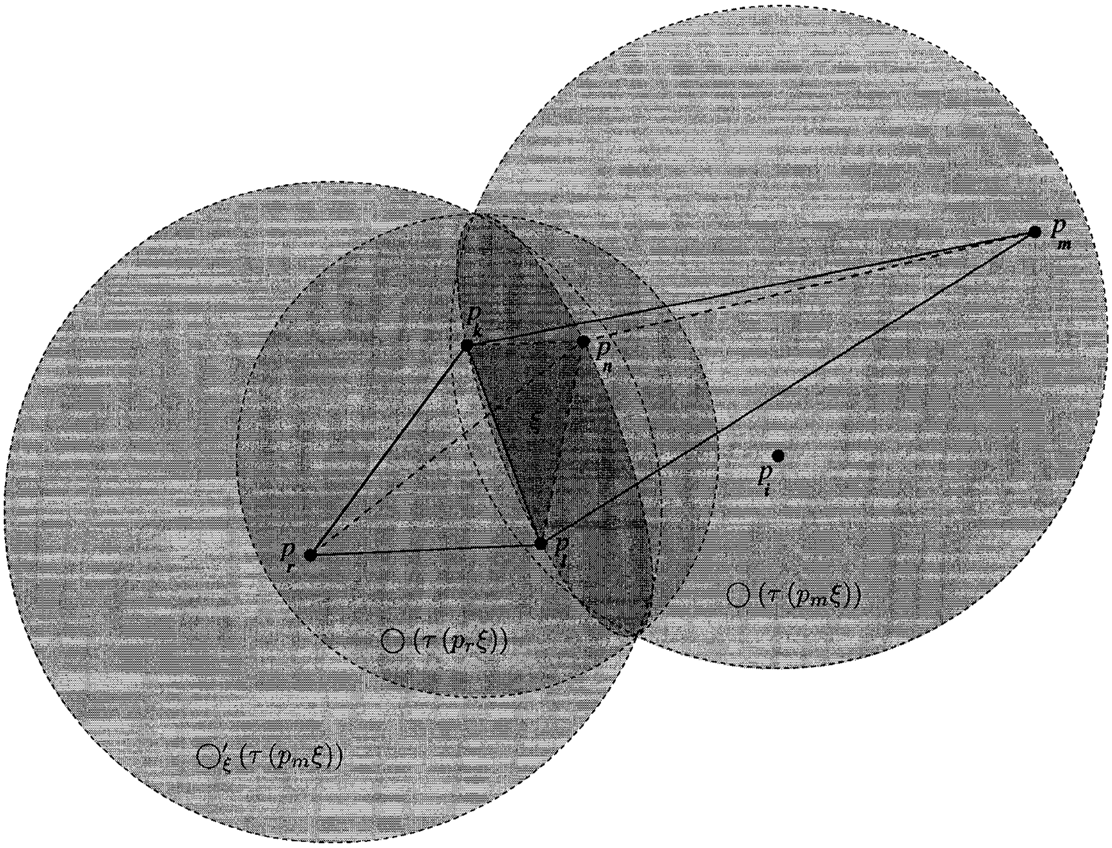


Figure 3.23: A face ξ at the boundary of the cavity of a Steiner point p_i, first case.

(i) If $r(\tau(p_m\xi)) > r(\tau(p_r\xi))$, see Figure 3.23, then

$$\Psi \subset \left(\bigcirc(\tau(p_m\xi)) \setminus \bigcirc'_\xi(\tau(p_m\xi))\right),$$

and, according to Lemma 3.8, $r(\tau(p_i\xi)) < r(\tau(p_m\xi))$.

(ii) If $r(\tau(p_m\xi)) \leq r(\tau(p_r\xi))$, see Figure 3.24, then

$$\Psi \subset \left(\bigcirc'_\xi(\tau(p_r\xi)) \setminus \bigcirc(\tau(p_r\xi))\right)$$

and, by Lemma 3.8, $r(\tau(p_i\xi)) < r(\tau(p_r\xi))$.

■

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

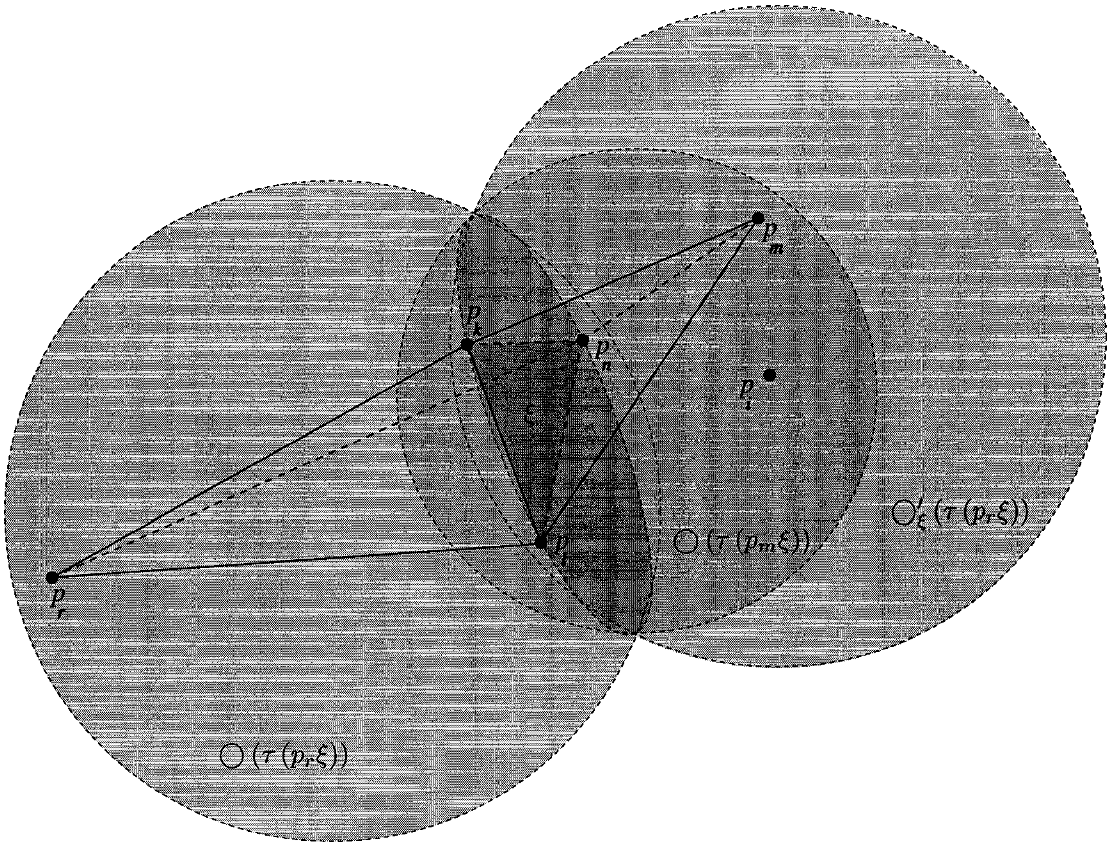


Figure 3.24: A face ξ at the boundary of the cavity of a Steiner point p_i, second case.

Lemma 3.10 (Sufficient condition of Delaunay-independence in 3D) *Points p_i and p_j are Delaunay-independent if there exists a subsegment s of segment $\mathcal{L}(p_ip_j)$ such that all tetrahedron circumdisks which intersect s have diameter less than or equal to the length of s, i.e.,*

$$\exists s \subseteq \mathcal{L}(p_ip_j) \ : \ \forall \tau \in T \ : \ s \cap \bigcirc(\tau) \neq \emptyset \implies 2r(\tau) \leq |s|, \tag{3.27}$$

where $|s|$ is the length of s.

Proof: First, condition (3.27) implies that $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$. Indeed, if there had been a tetrahedron circumdisk which included both p_i and p_j, then the diameter of this circumdisk would be greater than the length of $\mathcal{L}(p_ip_j)$ which would contradict (3.27).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

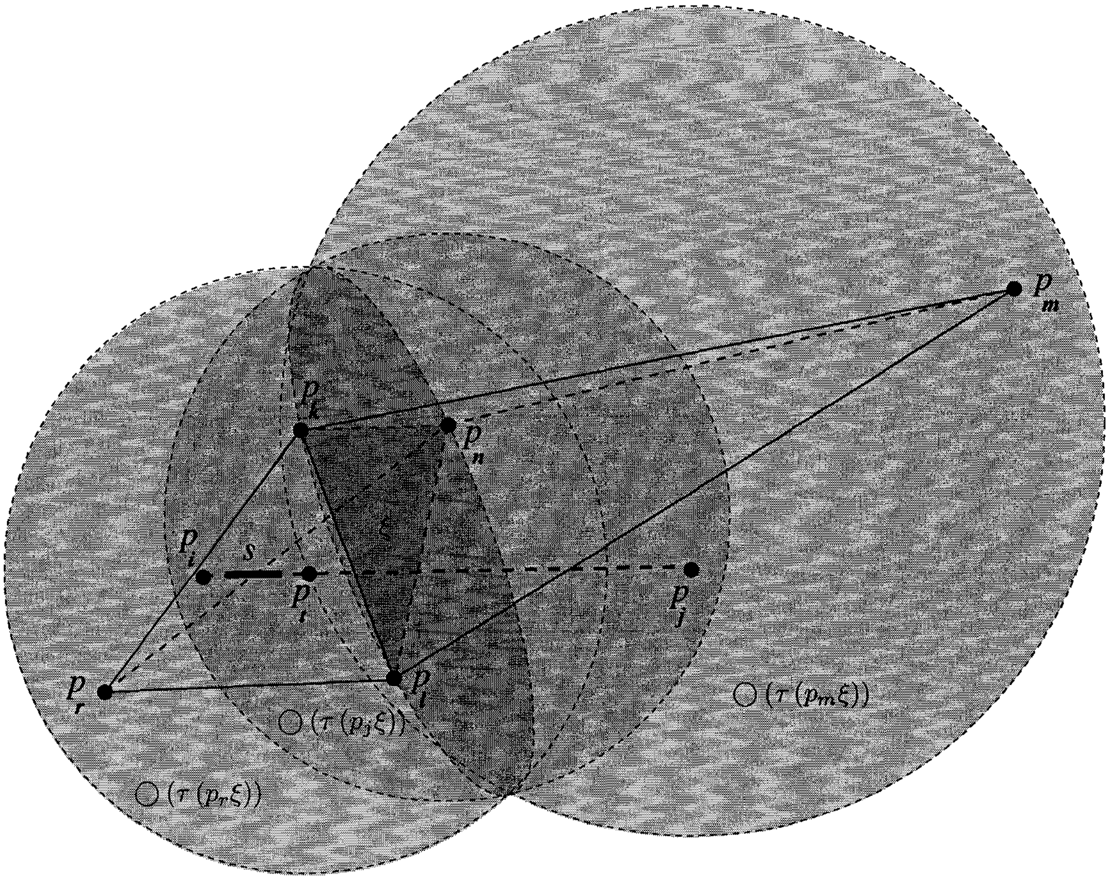


Figure 3.25: A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, first case.

Now, there are two possibilities:

(i) If $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) = \emptyset$, then, by Corollary 3.1, p_i and p_j are Delaunay-independent.

(ii) Otherwise, let $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) \neq \emptyset$ and $\xi = \triangle(p_k p_l p_n)$ be an arbitrary face in $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, such that $\tau(p_r\xi) \in \mathcal{C}(p_i)$ and $\tau(p_m\xi) \in \mathcal{C}(p_j)$. We are going to prove that $p_i \notin \bigcirc(\tau(p_j\xi))$ and, thus, p_i and p_j are Delaunay-independent by Theorem 3.1. The proof is by contradiction. Suppose condition (3.27) holds and $p_i \in \bigcirc(\tau(p_j\xi))$. There are two cases:

(ii-a) If $r(\tau(p_m\xi)) > r(\tau(p_r\xi))$, see Figure 3.25, then from Lemma 3.9 it follows

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

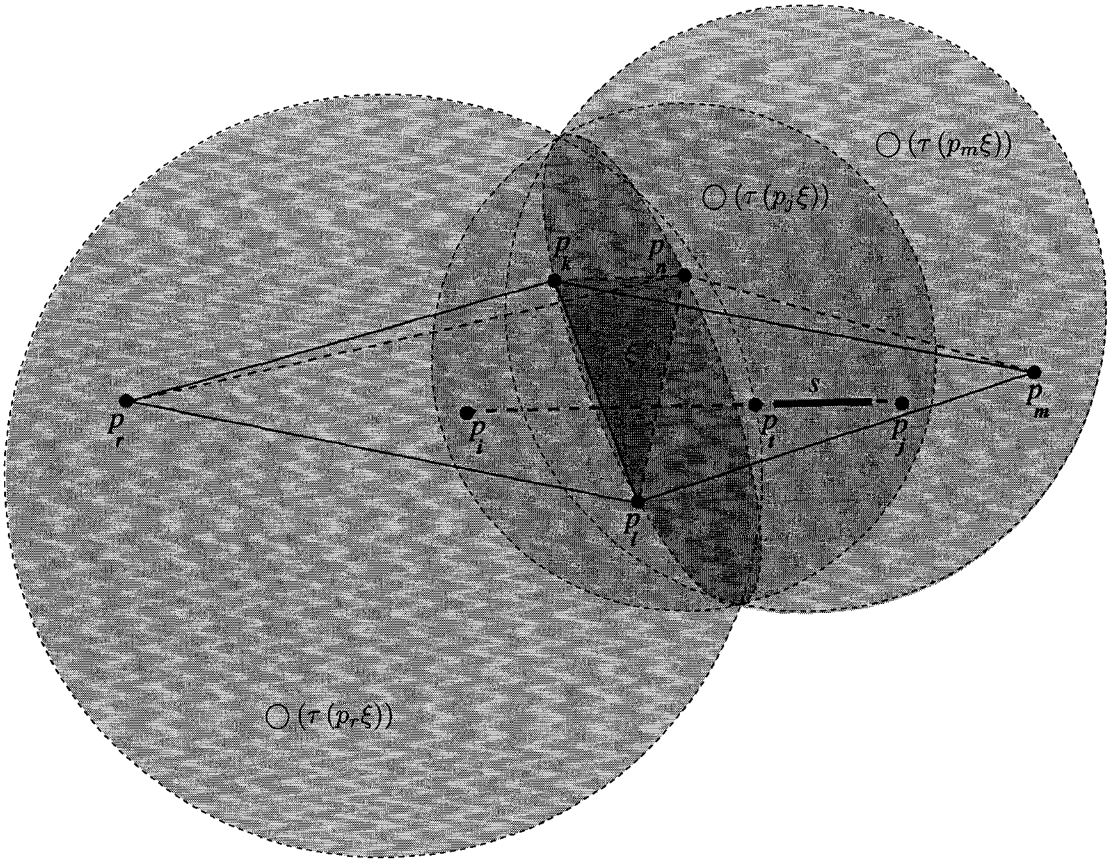


Figure 3.26: A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, second case.

that

$$r(\tau(p_j\xi)) < r(\tau(p_m\xi)). \tag{3.28}$$

In addition, the assumption that $p_i \in \bigcirc(\tau(p_j\xi))$ implies that

$$|\mathcal{L}(p_ip_j)| < 2r(\tau(p_j\xi)). \tag{3.29}$$

From (3.28) and (3.29) we conclude that the following relation holds:

$$|\mathcal{L}(p_ip_j)| < 2r(\tau(p_m\xi)). \tag{3.30}$$

Due to (3.30) and the assumption that (3.27) holds as well as the fact that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$|s| \leq |\mathcal{L}(p_i p_j)|$, we conclude that s cannot intersect $\bigcirc(\tau(p_m \xi))$. If p_t is the point of intersection of $\mathcal{L}(p_i p_j)$ with the boundary of $\bigcirc(\tau(p_m \xi))$, then s is restricted to be the subsegment of $\mathcal{L}(p_i p_t)$ and

$$|s| \leq |\mathcal{L}(p_i p_t)|. \tag{3.31}$$

From the assumptions that $p_i \in \bigcirc(\tau(p_j \xi))$ and $p_i \notin \bigcirc(\tau(p_m \xi))$, it follows that p_i has to lie in the region $\bigcirc(\tau(p_j \xi)) \setminus \bigcirc(\tau(p_m \xi))$ and the following two relations hold:

$$s \cap \bigcirc(\tau(p_r \xi)) \neq \emptyset \tag{3.32}$$

and

$$|\mathcal{L}(p_i p_t)| < 2r(\tau(p_r \xi)). \tag{3.33}$$

Relations (3.31), (3.32), and (3.33) together imply that the condition (3.27) does not hold and we have come to a contradiction.

(ii-b) If $r(\tau(p_m \xi)) \leq r(\tau(p_r \xi))$, see Figure 3.26, then from Lemma 3.9 it follows that $r(\tau(p_j \xi)) < r(\tau(p_r \xi))$ and considering that

$$|s| \leq |\mathcal{L}(p_i p_j)| < 2r(\tau(p_j \xi)) < 2r(\tau(p_r \xi))$$

we conclude that s cannot intersect $\bigcirc(\tau(p_r \xi))$. This limits s to lie within the subsegment $\mathcal{L}(p_j p_t)$, where p_t is the point of intersection of $\mathcal{L}(p_i p_j)$ with the boundary of $\bigcirc(\tau(p_r \xi))$; therefore,

$$|s| \leq |\mathcal{L}(p_j p_t)|. \tag{3.34}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The subsegment $\mathcal{L}\left(p_j p_t\right)$ lies completely inside the region

$$\bigcirc\left(\tau\left(p_j\xi\right)\right) \setminus \bigcirc\left(\tau\left(p_r\xi\right)\right)$$

which in turn is completely inside $\bigcirc\left(\tau\left(p_m\xi\right)\right)$, hence the following two relations hold:

$$s \cap \bigcirc\left(\tau\left(p_m\xi\right)\right) \neq \emptyset \tag{3.35}$$

and

$$\left|\mathcal{L}\left(p_j p_t\right)\right| < 2r\left(\tau\left(p_m\xi\right)\right). \tag{3.36}$$

Relations (3.34), (3.35), and (3.36) together imply that the condition (3.27) does not hold and we have come to a contradiction.

$\blacksquare$

3.4.2 Octree Construction

Let $\Lambda_x = \{\mathit{Left}, \mathit{Right}\}$, $\Lambda_y = \{\mathit{Top}, \mathit{Bottom}\}$, and $\Lambda_z = \{\mathit{Back}, \mathit{Front}\}$ be the possible directions of face-adjacent leaves of an octree.

Definition 3.7 (α-neighborhood) *Let the* α-neighborhood $\mathcal{N}_\alpha(L)$ of an octree leaf L $(\alpha \in \Lambda_x \cup \Lambda_y \cup \Lambda_z)$ *be the set of octree leaves that share a face with* L *and are located in the* α *direction of* L.

Definition 3.8 (Buffer zone) *Let the set of leaves*

$$\begin{aligned} \mathrm{BUF}(L) = \; & \bigcup_{\alpha\in\Lambda_x} \mathcal{N}_\alpha(L) \cup \\ & \bigcup_{\beta\in\Lambda_y} \{\mathcal{N}_\beta(L') \mid L' \in \mathcal{N}_\alpha(L)\} \cup \\ & \bigcup_{\gamma\in\Lambda_z} \{\mathcal{N}_\gamma(L'') \mid L'' \in \{\mathcal{N}_\beta(L') \mid L' \in \mathcal{N}_\alpha(L)\}\} \end{aligned}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

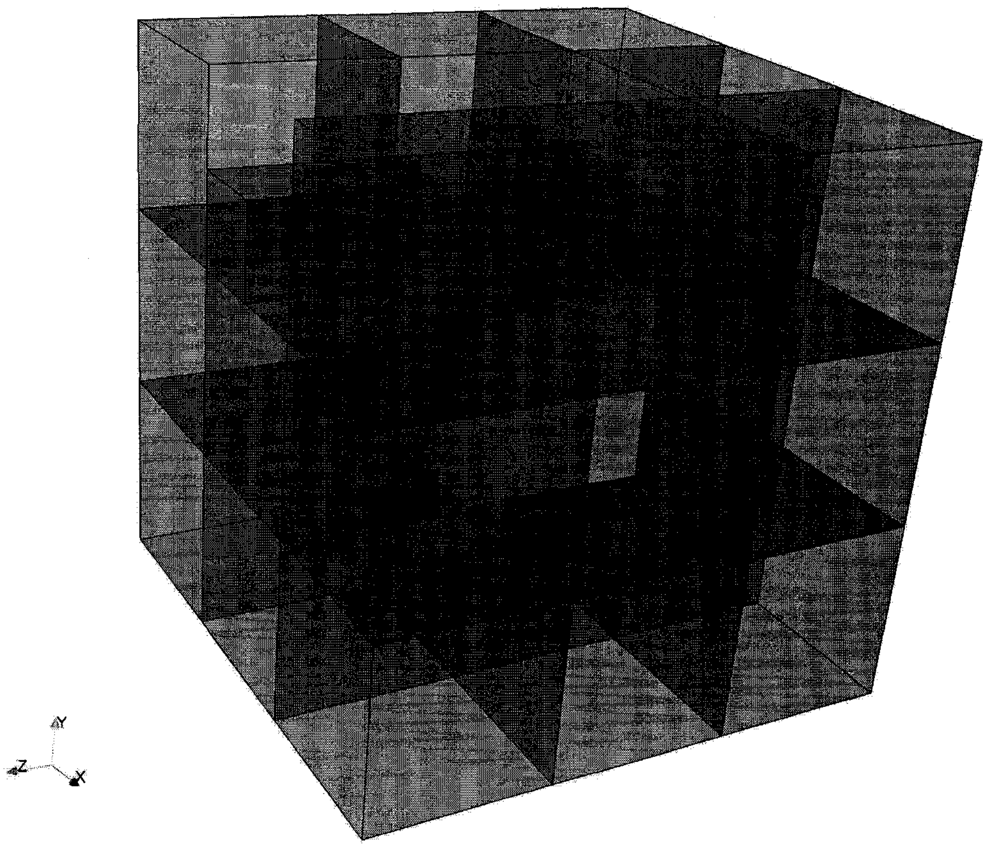


Figure 3.27: The buffer zone of an octree leaf L. The leaf L is the solid white box in the center. The transparent boxes around it is $\mathrm{BUF}(L)$.

under the condition

$$\forall L' \in \mathrm{BUF}(L), \forall \tau \in T \; : \; \bigcirc(\tau) \cap L' \neq \emptyset \implies r(\tau) < \frac{1}{6}\ell(L'), \tag{3.37}$$

be called a buffer zone *of leaf L with respect to mesh $\mathcal{M}$, where $\ell(L')$ is the length of the edge of cube L'.*

Lemma 3.11 (Sufficient condition of leaf Delaunay-separateness in 3D) *If L_i and L_j are octree leaves, $i \neq j$, and $L_j \notin \mathrm{BUF}(L_i)$, then L_i and L_j are Delaunay-separated.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

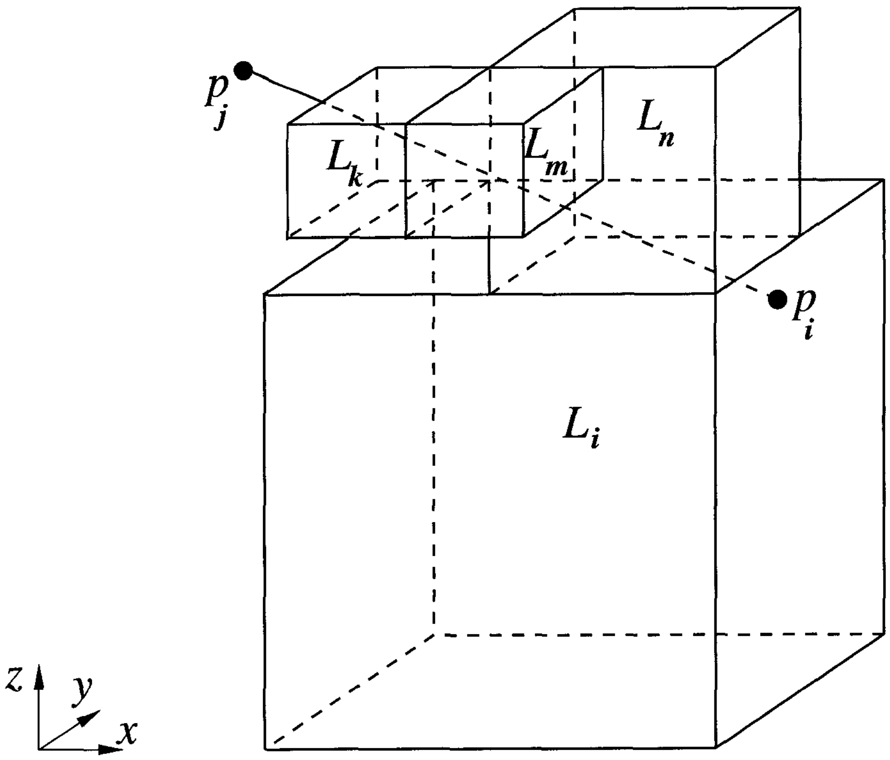


Figure 3.28: A line segment $\mathcal{L}(p_ip_j)$ which intersects buffer leaves L_n, L_m, $L_k \in$ BUF (L_i). The point p_i lies inside the leaf L_i, and the point p_j is outside any of the leaves from BUF (L_i).

Proof: We will start by proving that, for an arbitrary pair of points $p_i \in L_i$ and $p_j \in L_j \notin$ BUF (L_i), p_i and p_j are Delaunay-independent. Then we will extend the proof to show that any pair of points from $\{p_i, p_i', p_i''\} \times \{p_j, p_j', p_j''\}$ are Delaunay-independent, which will imply that p_i and p_j are strongly Delaunay-independent; hence, L_i and L_j are Delaunay-separated.

Let the coordinates of the points be $p_i = (x_i, y_i, z_i)$ and $p_j = (x_j, y_j, z_j)$. Without the loss of generality suppose $x_i > x_j$, $y_i > y_j$, $z_i < z_j$. Then $\mathcal{L}(p_ip_j)$ can enter the leaves it intersects by passing only through their bottom, back, and right faces, and it can exit the leaves by passing only through their top, front, and left faces.

Let $\mathcal{L}(p_ip_j)$ intersect the common surface of the top face of L_i with the bottom face of L_n, see Figure 3.28 (the case of intersection of $\mathcal{L}(p_ip_j)$ with the left and the front

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

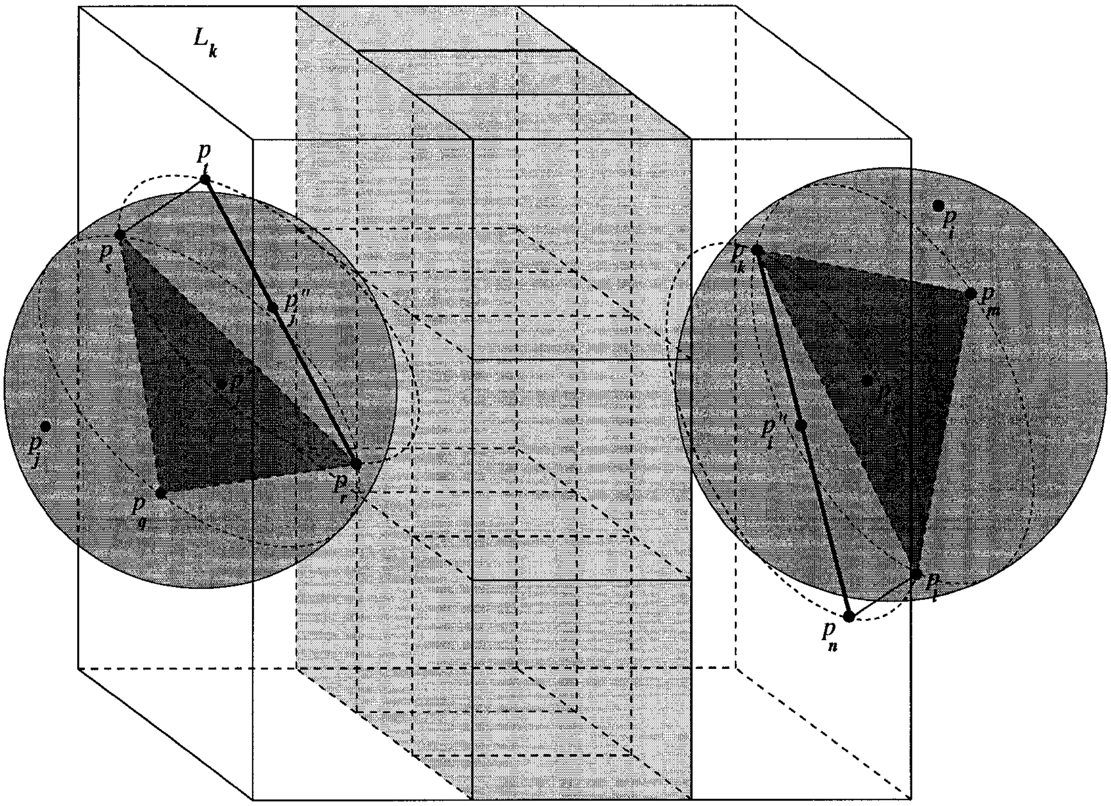


Figure 3.29: Leaf Delaunay-separateness.

faces of L_n is symmetric up to rotation). Below we consider the case when $\mathcal{L}(p_i p_j)$ exits through the top face of L_n, the case when it exits through the front face of L_n (the case when it exits through the left face is symmetric up to rotation), and all possible sub-cases. We show that in each case condition (3.27) is satisfied which means that p_i and p_j are Delaunay-independent by Lemma 3.10.

(i) If $\mathcal{L}(p_i p_j)$ exits through the top face of L_n, then the subsegment $s = \mathcal{L}(p_i p_j) \cap L_n$ has length greater than $\ell(L_n)$ by a simple projection argument. By the construction of the buffer zone, see Definition 3.8, all tetrahedron circumdisks which intersect L_n have diameter less than $\ell(L_n)$. Therefore, condition (3.27) is satisfied.

(ii) Otherwise, let $\mathcal{L}(p_i p_j)$ exit through the front face of L_n. Let $L_m \in \mathrm{BUF}(L_i)$ be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the leaf which is adjacent to L_n at the point of intersection with $\mathcal{L}(p_i p_j)$. Assume that $\ell(L_m) \leq \ell(L_n)$ (otherwise, change the order of L_m and L_n).

(ii-a) If $\mathcal{L}(p_i p_j)$ intersects the top face of L_m, then either

$$\max_z L_m = \max_z L_n$$

or

$$\max_z L_m < \max_z L_n,$$

by the requirement of octree conformity. If

$$\max_z L_m < \max_z L_n,$$

apply this argument to the leaf directly above L_m. Otherwise, let

$$\max_z L_m = \max_z L_n.$$

Then the subsegment $s = \mathcal{L}(p_i p_j) \cap (L_n \cup L_m)$ has length greater than $\ell(L_n)$ and all tetrahedron circumdisks which intersect s have diameter less than $\max\{\ell(L_m), \ell(L_n)\} = \ell(L_n)$, hence condition (3.27) is satisfied.

(ii-b) If $\mathcal{L}(p_i p_j)$ intersects the front face of L_m, then the subsegment $s = \mathcal{L}(p_i p_j) \cap L_m$ has length greater than $\ell(L_m)$, all tetrahedron circumdisks which intersect s have diameter less than $\ell(L_m)$, and condition (3.27) is satisfied.

(ii-c) If $\mathcal{L}(p_i p_j)$ intersects the left face of L_m, then let $L_k \in \mathrm{BUF}(L_i)$ be the leaf adjacent to L_m at the point of intersection with $\mathcal{L}(p_i p_j)$. Assume that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$\ell(L_k) \leq \ell(L_m)$ (otherwise, change the order of L_k and L_m). There are three sub-cases:

(ii-c-1) If $\mathcal{L}(p_ip_j)$ intersects the left face of L_k then the subsegment $s = \mathcal{L}(p_ip_j) \cap L_k$ has length greater than $\ell(L_k)$, all tetrahedron circumdisks which intersect s have diameter less than $\ell(L_k)$, and condition (3.27) is satisfied.

(ii-c-2) If $\mathcal{L}(p_ip_j)$ intersects the top face of L_k then either

$$\max_z L_k = \max_z L_m$$

or

$$\max_z L_k < \max_z L_m,$$

by the requirement of octree conformity. If

$$\max_z L_k < \max_z L_m,$$

apply this argument to the leaf directly above L_k. Otherwise, let

$$\max_z L_k = \max_z L_m.$$

Then the subsegment $s = \mathcal{L}(p_ip_j) \cap (L_n \cup L_m \cup L_k)$ has length greater than $\ell(L_n)$, all tetrahedron circumdisks which intersect s have diameter less than $\max\{\ell(L_n), \ell(L_m), \ell(L_k)\} = \ell(L_n)$, and condition (3.27) is satisfied.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(ii-c-3) If $\mathcal{L}(p_ip_j)$ intersects the front face of L_k then either

$$\min_y L_k = \min_y L_m$$

or

$$\min_y L_k > \min_y L_m,$$

by the requirement of octree conformity. If

$$\min_y L_k > \min_y L_m,$$

apply this argument to the leaf directly in front of L_k. Otherwise, let

$$\min_y L_k = \min_y L_m.$$

The subsegment $s = \mathcal{L}(p_ip_j) \cap (L_m \cup L_k)$ has length greater than $\ell(L_m)$, all tetrahedron circumdisks which intersect s have diameter less than $\max\{\ell(L_m), \ell(L_k)\} = \ell(L_m)$, and condition (3.27) is satisfied.

Now suppose $p_i \in L_i$, $p_j \in L_j \notin \mathrm{BUF}(L_i)$, and $L_k \in \mathrm{BUF}(L_i)$, see Figure 3.29. We will consider the worst case, i.e., the case when two points, one from $\{p_i, p_i', p_i''\}$ and the other from $\{p_j, p_j', p_j''\}$, can be as close as possible, which happens when p_i', p_i'', p_j', and p_j'' are inside L_k. Suppose p_i encroaches upon a constrained face $\triangle(p_kp_lp_m)$, i.e., p_i is inside the open equatorial disk $\bigcirc(\triangle(p_kp_lp_m))$. Then the Delaunay refinement algorithm will reject p_i and attempt to insert the circumcenter p_i' of $\triangle(p_kp_lp_m)$. Furthermore, suppose p_i' encroaches upon a constrained segment $e(p_np_k)$, i.e., p_i' is inside the open diametral

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

disk $\bigcirc(e(p_n p_k))$. Similarly, let p_j encroach upon a constrained face $\triangle(p_q p_r p_s)$ and p_j' encroach upon a constrained segment $e(p_r p_t)$.

We have:

$$\begin{array}{lll} \|p_i - p_i'\| & < r(\triangle(p_k p_l p_m)) & (\text{since } p_i \in \bigcirc(\triangle(p_k p_l p_m))) \\ & < \frac{1}{6}\ell(L_k) & (\text{follows from (3.37) since } p_i' \in L_k). \end{array} \tag{3.38}$$

Furthermore,

$$\begin{array}{lll} \|p_i' - p_i''\| & < r(e(p_n p_k)) & (\text{since } p_i' \in \bigcirc(e(p_n p_k))) \\ & < \frac{1}{6}\ell(L_k) & (\text{follows from (3.37) since } p_i'' \in L_k). \end{array} \tag{3.39}$$

From (3.38), (3.39), and the triangle inequality it follows that $\|p_i - p_i''\| < \frac{1}{3}\ell(L_k)$. By analogy, $\|p_j - p_j''\| < \frac{1}{3}\ell(L_k)$. Finally, since p_i and p_j are separated by L_k, we conclude that $\|p_i'' - p_j''\| > \frac{1}{3}\ell(L_k)$. By constructing imaginary buffer cubes $\{L_{k_\eta}\}$ ($\eta = 1, \ldots, 9$) inside L_k, as shown in Figure 3.29, we can still satisfy condition (3.37) for each of L_{k_η}, which guarantees that p_i'' and p_j'' are Delaunay-independent by Lemma 3.10. ■

3.4.3 Tetrahedron Quality Measures

In two dimensions, the most commonly used measures of element quality are minimal angle, aspect ratio, and circumradius-to-shortest edge ratio. The aspect ratio is usually defined as the ratio of the radius of the smallest circumscribed sphere to the radius of the largest inscribed sphere [5]. In two dimensions, these metrics are equivalent, i.e., an improvement of one of the criteria implies an improvement of the others. In three dimensions, however, the situation is more complicated since such correspondence does not hold. The problem is illustrated by the existence of so-called *sliver* elements, see Figure 3.30. A sliver is formed by positioning the three vertices of a tetrahedron's base

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

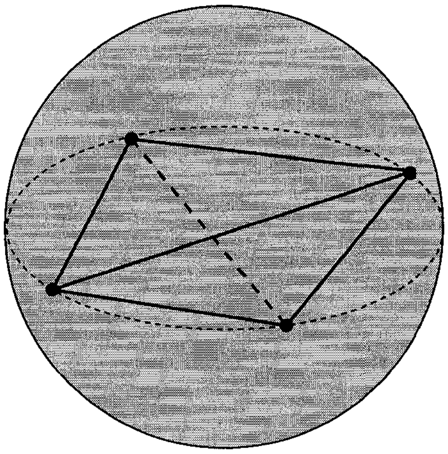


Figure 3.30: A sliver tetrahedron.

along the equatorial circle of a sphere and moving the apex vertex arbitrarily close to the plane of the base. Obviously, such a tetrahedron can have very good circumradius-to-shortest edge ratio since all the edges can have lengths comparable to the radius of the sphere. However, the aspect ratio can be made arbitrarily high due to the fact that the radius of the inscribed sphere approaches zero as the apex approaches the plane of the base.

Delaunay refinement algorithm in both two and three dimensions improves the circumradius-to-shortest edge ratio. Unfortunately, in three dimensions slivers can survive. Recently, an algorithm was developed for the construction of sliverless meshes by weighted Delaunay refinement [11]. However, it can offer only a very small angle guarantee made possible by the weight pumping method. Below we will focus only on the standard (unweighted) Delaunay refinement algorithm. Although slivers can impose a problem for some numerical methods, it has been shown in [63, 64] that bounded circumradius-to-shortest edge ratio of mesh elements is sufficient to obtain optimal convergence rates for the solution of Poisson equation using the control volume method

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(CVM).

The following theorems relate the error estimates for the CVM with the quality measures of mesh elements. Let A_{ij} be the Voronoi face associated with the Delaunay edge e_{ij}. Then for the quantities defined on the edges of the Delaunay mesh, the inner product is defined in [64, 67] as

$$(U, V)_W = \sum_{(i,j)} |A_{ij}||e_{ij}|U_{ij}V_{ij},$$

where $\|U\|_W$ is a discrete version of the $L_2(\Omega)$ norm of the gradient.

Theorem 3.6 (Nicolaides [67]) *Let $\{u_i\}$ be the discrete solution given by the control volume method, and define $U_{ij} = (u_i - u_j)/h_{ij}$ and $U_{ij}^{(1)}$ and $U_{ij}^{(2)}$ by*

$$U_{ij}^{(1)} = \frac{u(p_i) - u(p_j)}{h_{ij}}, \qquad U_{ij}^{(2)} = \frac{1}{|A_{ij}|}\int_{A_{ij}} -\frac{\partial u}{\partial n}$$

(u is the exact solution), then $\|U - U^{(1)}\|_W \leq \|U^{(1)} - U^{(2)}\|_W$.

Theorem 3.7 (Miller et al. [64]) *Let $\rho = \max_{(i,j)} r_{ij}/|e_{ij}|$ be the radius-edge aspect ratio of a Delaunay triangulation of Ω, then*

$$\|U^{(1)} - U^{(2)}\|_W \leq (1 + 4\rho^2)\|D^2 u\|_{L^2(\Omega)} h, \tag{3.40}$$

where $D^2 u$ is the Hessian (matrix of second derivatives) of u, and $h = \max_{(i,j)} |e_{ij}|$.

The error estimation (3.40) implies that there are two mesh related parameters which can be adjusted to improve the accuracy of the solution and are within the control of a mesh generation algorithm: the upper bound $\bar{\rho}$ on the circumradius-to-shortest edge

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ratio, and the length of the longest edge h. Since the length of the longest edge is bounded by twice the circumradius of a tetrahedron, for the development of our three-dimensional parallel Delaunay refinement algorithm we chose to use the tetrahedron's circumradius as the measure of element size. In contrast to the two-dimensional case, we do not use the volume of the elements.

3.4.4 Implementation and Evaluation

For the implementation of our parallel Delaunay refinement algorithm we were able to utilize the serial Delaunay refinement code realized in `Tetgen` [78–80]. We consider this as a major accomplishment towards the goal of separating the parallel and the sequential design issues in parallel mesh generation. Indeed, `Tetgen` consists of about 33 thousand lines of highly optimized C++ code which took its author (Hang Si) several years to write. In addition, the implementation is based on a large number of theoretical and algorithmic results which were published during the last several decades and keep being introduced. Therefore, it is imperative that the development of the sequential part of the software be separated from the parallel part.

Figure 3.31 presents a high level diagram of our software design. The blocks marked "Serial Delaunay Refinement" represent P instances of sequential Delaunay refinement code which is `Tetgen` in our implementation, but could be another code as well, e.g., `Pyramid` [75], `Tetmesh` [84], or `Gridgen` [41]. The "Element Scheduling" boxes represent the management of poor quality element queues by the sequential code. In our implementation, we modified `Tetgen` to create a separate queue for each of the leaves of the octree, and since each leaf at a given time can be refined by only one thread, each thread

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

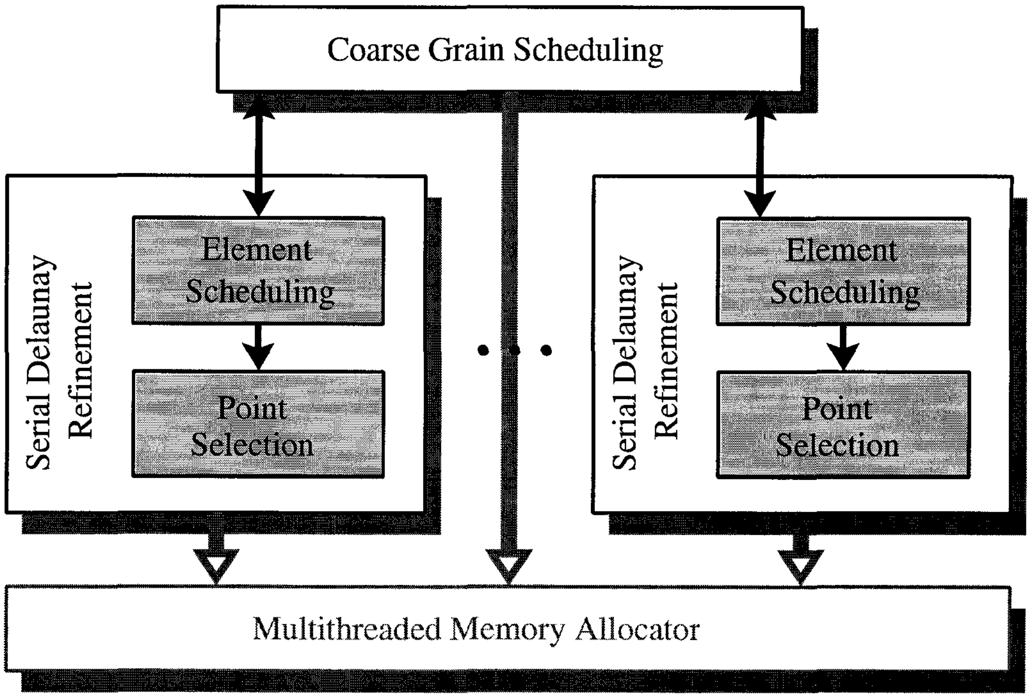


Figure 3.31: The diagram of the design of our parallel Delaunay refinement software.

pops from and pushes into a separate poor element queue. The "Point Selection" box is the abstraction for choosing a particular point insertion strategy inside the circumdisk of a poor quality element. As we have shown in Section 2.3, sequential Delaunay refinement algorithms have the flexibility to choose Steiner points from the regions inside the circumdisk of a poor quality tetrahedron which we call the selection disks.

The box marked "Coarse Grained Scheduling" represents the construction of the octree and the scheduling of leaves for the refinement. The leaves with larger volumes have higher refinement priorities than the leaves with smaller volumes, and the leaves of the same size are processed in the first-in-first-out order. This simple strategy is designed to achieve maximal concurrency as fast as possible without introducing large overheads. The development of more efficient scheduling algorithms may be the topic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

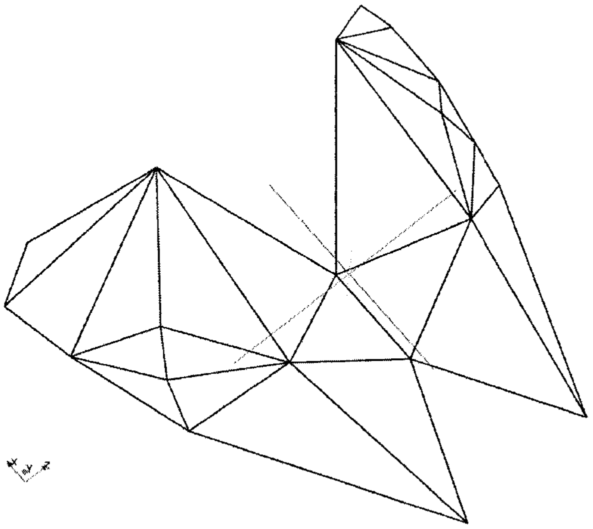


Figure 3.32: A wireframe model of a flying bat.

of future research.

For the multithreaded memory allocator, we have experimented with several approaches, see [14] for a detailed description and performance evaluation. The best results from the performance point of view have been achieved with the use of a custom designed memory allocator. The standard memory allocator backing `new` and `delete` on our experimental platforms imposed unacceptable overhead, which motivated the development of a custom allocation scheme. At initialization, the memory pool class takes the size of the underlying object as a parameter and at runtime it allocates blocks of memory which can accommodate a certain number of objects. The memory pool class does not address such issues as aligning the objects to memory addresses and choosing appropriate block sizes. The design of a more general and sophisticated memory manager was addressed in [71].

Figure 3.32 shows a wireframe model of a flying bat used in the simulation and visualization of air flow around bat wings [69]. The modeling is performed by constructing a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

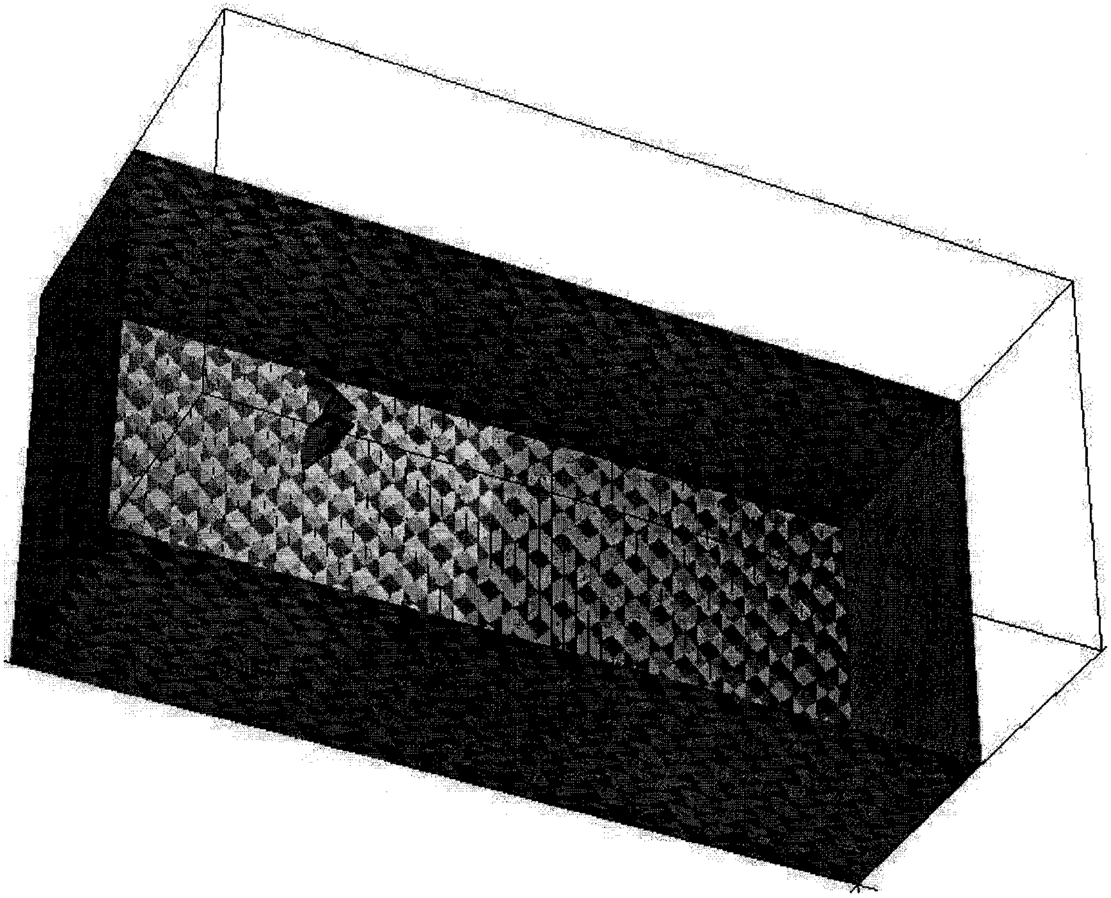


Figure 3.33: The bat inside a box.

large box containing the bat, see Figure 3.33. A tetrahedral mesh is constructed in the interior of the box, such that the face ahead of the bat is considered the inflow, the face behind is the outflow, and the other faces are paired for the use of periodic boundary conditions. The mesh is used as input to the spectral/hp element solver Nektar [89] which solves the incompressible Navier-Stokes equations in arbitrary Lagrangian-Eulerian formulation.

The most interesting physical phenomena like high vorticity happen in the area directly adjacent to the bat and in the trail just behind it. That is why these areas require a more refined mesh to capture their details. We defined a second box of parameterizable

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

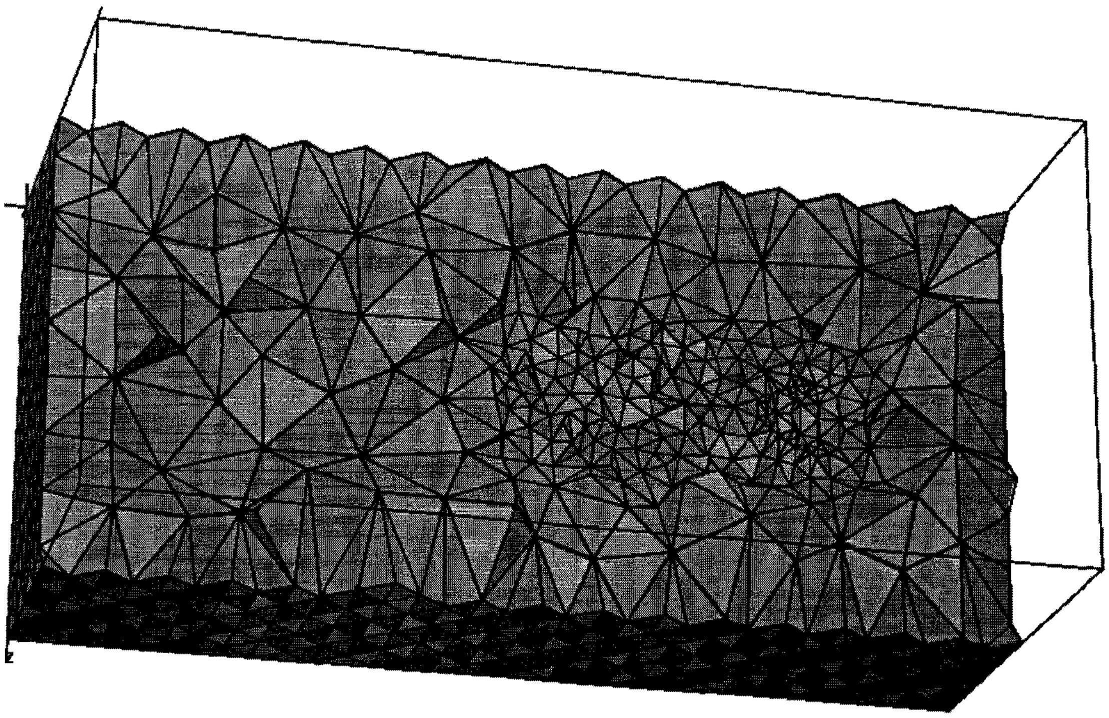


Figure 3.34: A nonuniform mesh of the bat inside a box.

Table 3.5: The total time spent by the three-dimensional PGDR code on refining the meshes of the unit cube and the bat models. For the unit cube, $\bar{r} = 0.015$ and the final mesh size is 1.89 million. For the bat $\bar{r} = 0.25$ if $(-19.89 < x < 6.50) \wedge (-5.65 < y < 8.05) \wedge (-5.61 < z < 5.61)$; and $\bar{r} = 0.5$ otherwise; 5.8 million tetrahedra.

Model	Number of compute threads			
	1	2	3	4
Unit cube	77.2	47.0	37.06	32.29
Bat	235.2	142.1	120.0	111.2

size inside the large box to specify the area of fine refinement, see Figure 3.34.

We ran our experiments on the SciClone Cluster at the College of William and Mary and used one of its "Vortex" nodes which is a quad-cpu Sun Fire V440 server with 1.28 GHz clock and 8 GB of main memory. Table 3.5 summarizes the running times of our experiments.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Conclusions

4.1 Summary and Extensions

We analyzed the existing point insertion methods for guaranteed quality Delaunay refinement and unified them into a framework which allows to develop customized mesh optimization techniques. The goals of these techniques may include the following:

- minimizing the number of inserted points, see for example [86, 87] and Subsections 2.2.2 and 2.3.2 here;
- eliminating slivers, see [25];
- splitting multiple poor quality triangles simultaneously, see Figure 4.1(left).
- creating elongated edges in required directions, see Figure 4.1(center);
- inserting more than one point simultaneously, e.g., to create elements with specific shapes, see Figure 4.1(right);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Parallel Delaunay meshing methods / ***Properties***	Stability	No domain decomposition	No fine grain synchronization	No rollbacks	Extended to 3D	Code reuse
Parallel Optimistic Delaunay Meshing [27, 66]	●	●			●	
Parallel Generalized Delaunay Refinement [15–21, 29]	●	●	●	●	●	●
Parallel Constrained Delaunay Meshing [22]	●		●	●		
Parallel Delaunay Refinement Mesh Generation [46–48]	●	●				
Parallel Domain Delaunay Decoupling [58–60]	●		●	●		●

Table 4.1: A qualitative comparison of practical parallel Delaunay mesh generation methods.

- satisfying other application-specific requirements, for example, conformity to a boundary zone, see Figure 1.1.

The experiments with a new optimization-based two-dimensional point placement method show that it allows to improve the size of the mesh by up to 20% and up to 5% over the circumcenter and the off-center methods, respectively.

An extension of the selection disks to anisotropic mesh generation requires additional analysis. Labelle and Shewchuk [55] presented an anisotropic guaranteed-quality mesh generation algorithm. With each point p in Ω they associate a symmetric positive definite metric tensor which specifies how distances and angles are measured from the perspective of p. As a result, the Voronoi diagram of a point set becomes very complicated, and may even contain disconnected faces; therefore, it does not always dualize to a correct triangulation. The point insertion scheme developed in [55] takes into account the visibility of points with respect to Voronoi faces, which would also restrict the shape of a selection region.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

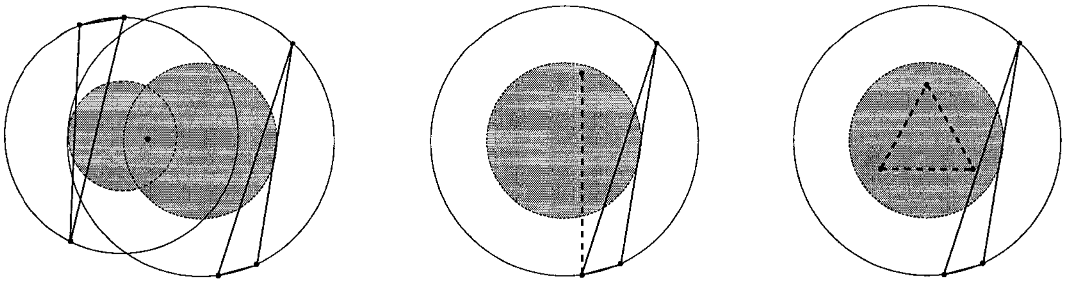


Figure 4.1: Examples of the approaches for choosing Steiner points within selection disks of skinny triangles.

We presented a theoretical framework for the development of parallel guaranteed quality Delaunay refinement codes for the construction of uniform and graded meshes, both in two and three dimensions. We eliminated such disadvantages of the previously proposed methods as the necessity to maintain a cavity (conflict) graph, the rollbacks, the requirement to solve a difficult domain decomposition problem, and the centralized sequential resolution of potential conflicts. The presented theory leverages the quality guarantees of the existing sequential Delaunay refinement algorithms. The algorithms are designed to work with custom point placement techniques which choose points from the selection disks. Currently they are limited to deterministic point selection; incorporating randomized point selection is left to the future research.

The parallelization of two-dimensional mesh generation algorithms is particularly important for some three-dimensional simulations which use multiple two-dimensional meshes in different coordinate systems. Some examples include direct numerical simulations of turbulence in cylinder flows with very high Reynolds numbers, see [34], and coastal ocean modeling for predicting storm surge and beach erosion in real-time, see [88]. For the modeling of turbulence, as shown by Karniadakis and Orszag [49],

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

with the increase of the Reynolds number Re, the size of the mesh grows in the order of $Re^{9/4}$, which motivates the use of parallel mesh generation algorithms. At the same time, the size of the mesh should be as small as possible given the required element quality constraints, which can be attained by using a nonuniform (graded) mesh.

In this dissertation we presented two types of Parallel Delaunay Refinement algorithms: uniform and graded. The sufficient condition for the uniform algorithm is based on the distance between the points which is similar to the definition of well-separated point sets in [9, 10]. However, the construction of the binary tree may be too expensive in practice, because all we really need is a uniform lattice. Our second type of algorithms (graded) does not involve a sufficient condition based on the distance between the points. There may be a non-trivial extension of our algorithms, so that they could be used in conjunction with the analysis in [9, 10], which we might look into in the future.

The uniform algorithm which we presented in Section 3.2 has been implemented both for shared- and distributed-memory architectures. The experiments on both architectures (see Table 3.4) suggest that the message passing overheads are not significant. The experimental evaluation of the uniform algorithm on distributed memory shows excellent scalability which is close to linear with respect to the number of processors. Another factor contributing to good scalability is that static work decomposition works well in this case. In particular, this leads to two advantages. First, we can use the fact that the amount of computation per area unit inside the domain is approximately equal for each of the blocks (subdomains). As a result, the amount of work assigned to different processors can be fairly well balanced. To process the domains of highly irregular shapes, one can use the approach based on overdecomposition and load balancing [4].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Second, the bookkeeping related to the background data structure (uniform lattice) is not very expensive compared to a quadtree used for a graded parallel Delaunay refinement algorithm. The uniform algorithm is therefore preferred for applications which do not require mesh gradation and at the same time need very large meshes which cannot fit into the memory of a single machine.

The graded algorithm which we presented in Section 3.3 for two dimensions and in Section 3.4 for three dimensions has been implemented for shared memory only. It is more suitable for applications which require the construction of the meshes with variable element size. The extension of this algorithm for distributed memory architectures is the subject of our future research. One of the questions which will need to be addressed is the development of the distributed quadtree (octree) data structure. One of the possible approaches has been described in [85], where each processor manages a local instance of the global octree with the use of a locational code lookup table for remote data accesses. As discussed in Section 3.4.4, the design of our algorithm on the high level can be viewed as the interaction of several components. The top-level component in Figure 3.31 marked as "Coarse Grain Scheduling" is the abstraction for the management of the quadtree (octree) and it will encapsulate the extensions to our existing implementation.

4.2 Publications

The following is a list of publications related to this dissertation:

- In the area of Computational Geometry and specifically Mesh Generation: SIAM J. Sci. Comp. [21], IMR'07 [23], IMR'06 [19], IMR'05 [18], SIAM J.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Comp. [15], CGW'06 [20], CGW'04 [16], USACM MeshTrends'07 [92], USACM MeshTrends'03 [13].

- In the area of Parallel and Distributed Computing: ACM ICS'05 [3], ACM ICS'04 [17], IEEE TPDS [4], JPDC [1,2], IEEE IPDPS'06 [52], IEEE IDAACS'05 [51], IMACS'05 [53], ICOSAHOM'04 [29].
- In the area of the Development of Mathematical Software: ACM TOMS [22], MATCOM [44], ICNGG'07 [14], ICNGG'05 [43], ICNGG'02 [12].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

[1] CHRISTOS D. ANTONOPOULOS, FILIP BLAGOJEVIC, ANDREY N. CHERNIKOV, NIKOS P. CHRISOCHOIDES, AND DIMITRIS S. NIKOLOPOULOS. Algorithm software and hardware optimizations for Delaunay mesh generation on simultaneous multithreaded architectures. *Journal on Parallel and Distributed Computing.* In revision, May 2007.

[2] CHRISTOS D. ANTONOPOULOS, FILIP BLAGOJEVIC, ANDREY N. CHERNIKOV, NIKOS P. CHRISOCHOIDES, AND DIMITRIS S. NIKOLOPOULOS. A multigrain Delaunay mesh generation method for multicore SMT-based architectures. *Journal on Parallel and Distributed Computing.* Submitted in Aug. 2006.

[3] CHRISTOS D. ANTONOPOULOS, XIAONING DING, ANDREY N. CHERNIKOV, FILIP BLAGOJEVIC, DIMITRIS S. NIKOLOPOULOS, AND NIKOS P. CHRISOCHOIDES. Multigrain parallel Delaunay mesh generation: Challenges and opportunities for multithreaded architectures. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 367–376, Cambridge, MA, 2005. ACM Press.

[4] KEVIN BARKER, ANDREY CHERNIKOV, NIKOS CHRISOCHOIDES, AND KESHAV PINGALI. A load balancing framework for adaptive and asynchronous applications. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):183–192, February 2004.

[5] MARSHALL WAYNE BERN AND DAVID EPPSTEIN. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, Ding-Zhu Du and Frank Hwang, editors, Lecture Notes Series on Computing, pages 23–90. World Scientific, 1992.

[6] G. E. BLELLOCH, J.C. HARDWICK, G. L. MILLER, AND D. TALMOR. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.

[7] G. E. BLELLOCH, G. L. MILLER, AND D. TALMOR. Developing a practical projection-based parallel Delaunay algorithm. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 186–195, Philadelphia, PA, May 1996.

[8] ADRIAN BOWYER. Computing Dirichlet tesselations. *Computer Journal*, 24:162–166, 1981.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[9] PAUL B. CALLAHAN AND S. RAO KOSARAJU. Algorithms for dynamic closest pair and n-body potential fields. In *Proceedings of the 6th annual ACM-SIAM symposium on Discrete algorithms*, pages 263–272, San Francisco, CA, 1995.

[10] PAUL B. CALLAHAN AND S. RAO KOSARAJU. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[11] S.-W. CHENG, T. K. DEY, AND T. RAY. Weighted Delaunay refinement for polyhedra with small angles. In *Proceedings of the 14th International Meshing Roundtable*, pages 325–342, San Diego, CA, September 2005. Springer.

[12] ANDREY CHERNIKOV, KEVIN BARKER, AND NIKOS CHRISOCHOIDES. Parallel programming environment for mesh generation. In *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 805–814, Honolulu, HI, June 2002.

[13] ANDREY CHERNIKOV AND NIKOS CHRISOCHOIDES. Automatic domain decomposition for parallel 2d constrained Delaunay mesh generation. In *4th Symposium on Trends in Unstructured Mesh Generation*, Albuquerque, NM, July 2003. Available online at http://www.andrew.cmu.edu/user/sowen/usnccm03/agenda.html.

[14] ANDREY N. CHERNIKOV, CHRISTOS D. ANTONOPOULOS, NIKOS P. CHRISOCHOIDES, SCOTT SCHNEIDER, AND DIMITRIOS S. NIKOLOPOULOS. Experience with memory allocators for parallel mesh generation on multicore architectures. Accepted to the 10th ISGG Conference on Numerical Grid Generation, Forth, Crete, Greece, 2007.

[15] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES. Parallel generalized Delaunay mesh refinement. Revision submitted to SIAM Journal on Computing in June 2007.

[16] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES. Parallel guaranteed quality planar Delaunay mesh generation by concurrent point insertion. In *Proceedings of the 14th Annual Fall Workshop on Computational Geometry*, pages 55–56, Cambridge, MA, November 2004. Electronic proceedings published at http://cgw2004.csail.mit.edu/proceedings.pdf.

[17] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES. Practical and efficient point insertion scheduling method for parallel guaranteed quality Delaunay refinement. In *Proceedings of the 18th Annual International Conference on Supercomputing*, pages 48–57, Malo, France, 2004. ACM Press.

[18] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES. Parallel 2D graded guaranteed quality Delaunay mesh refinement. In *Proceedings of the 14th International Meshing Roundtable*, pages 505–517, San Diego, CA, September 2005. Springer.

[19] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES. Generalized Delaunay mesh refinement: From scalar to parallel. In *Proceedings of the 15th International Meshing Roundtable*, pages 563–580, Birmingham, AL, September 2006. Springer.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[20] Andrey N. Chernikov and Nikos P. Chrisochoides. Parallel graded generalized Delaunay mesh refinement. In *Proceedings of the 16th Annual Fall Workshop on Computational Geometry*, Northampton, MA, November 2006. Electronic proceedings published at http://maven.smith.edu/~streinu/FwCG/proceedings.html.

[21] Andrey N. Chernikov and Nikos P. Chrisochoides. Parallel guaranteed quality Delaunay uniform mesh refinement. *SIAM Journal on Scientific Computing*, 28:1907–1926, 2006.

[22] Andrey N. Chernikov and Nikos P. Chrisochoides. Algorithm 872: Parallel 2D constrained Delaunay mesh generation. *ACM Transactions on Mathematical Software*, 34(1), March 2007. In press.

[23] Andrey N. Chernikov and Nikos P. Chrisochoides. Three-dimensional generalized Delaunay mesh refinement. Submitted to the 16th International Meshing Roundtable, Seattle, WA, 2007.

[24] L. Paul Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 274–280, San Diego, CA, 1993.

[25] L. Paul Chew. Guaranteed-quality Delaunay meshing in 3D. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 391–393, Nice, France, 1997.

[26] L. Paul Chew, Nikos Chrisochoides, and Florian Sukup. Parallel constrained Delaunay meshing. In *ASME/ASCE/SES Summer Meeting, Special Symposium on Trends in Unstructured Mesh Generation*, pages 89–96, Northwestern University, Evanston, IL, 1997.

[27] Nikos Chrisochoides and Démian Nave. Parallel Delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58:161–176, 2003.

[28] Nikos P. Chrisochoides. A survey of parallel mesh generation methods. Technical Report BrownSC-2005-09, Brown University, 2005. Also appears as a chapter in Numerical Solution of Partial Differential Equations on Parallel Computers (eds. Are Magnus Bruaset and Aslak Tveito), Springer, 2006.

[29] Nikos P. Chrisochoides, Andrey N. Chernikov, Andriy Kot, and Andriy Y. Fedorov. Parallel mesh generation: Web-services and COTS software. In *6th International Conference On Spectral and High Order Methods*, Providence, RI, 2004.

[30] Tim Culver. *Computing the Medial Axis of a Polyhedron Reliably and Efficiently.* PhD thesis, The University of North Carolina at Chapel Hill, 2000.

[31] Hugues L. de Cougny, Mark S. Shephard, and Can Ozturan. Parallel three-dimensional mesh generation. *Computing Systems in Engineering*, 5:311–323, 1994.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[32] Boris N. Delaunay. Sur la sphere vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Mataematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[33] Edsger W. Dijkstra. Shmuel Safra's version of termination detection. Circulated privately, http://www.cs.utexas.edu/users/EWD/ewd09xx/EWD998.PDF, January 1987.

[34] Suchuan Dong, Didier Lucor, and George Em Karniadakis. Flow past a stationary and moving cylinder: DNS at Re=10,000. In *Proceedings of the 2004 Users Group Conference (DOD_UGC'04)*, pages 88–95, Williamsburg, VA, 2004.

[35] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge University Press, England, 2001.

[36] Herbert Edelsbrunner and Damrong Guoy. Sink-insertion for mesh improvement. In *Proceedings of the 17th ACM Symposium on Computational Geometry*, pages 115–123, Medford, MA, 2001.

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

[38] J. Galtier and P. L. George. Prepartitioning as a way to mesh subdomains in parallel. In *Proceedings of the 5th International Meshing Roundtable*, pages 107–121, Pittsburgh, PA, 1996.

[39] Paul-Louis George and Houman Borouchaki. *Delaunay Triangulation and Meshing. Application to Finite Elements.* HERMES, 1998.

[40] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[41] Gridgen. http://www.pointwise.com/gridgen/index.shtml. Accessed on Apr. 21, 2004.

[42] Martin Isenburg, Yuanxin Liu, Jonathan Shewchuk, and Jack Snoeyink. Streaming computation of Delaunay triangulations. *ACM Transactions on Graphics*, 25(3):1049–1056, 2006.

[43] Y. Ito, A. M. Shih, A. K. Erukala, B. K. Soni, A. N. Chernikov, N. P. Chrisochoides, and K. Nakahashi. Generation of unstructured meshes in parallel using an advancing front method. In *Proceedings of the 9th International Conference on Numerical Grid Generation in Computational Field Simulations*, San Jose, CA, June 2005.

[44] Y. Ito, A. M. Shih, A. K. Erukala, B. K. Soni, A. N. Chernikov, N. P. Chrisochoides, and K. Nakahashi. Parallel mesh generation using an advancing front method. *Mathematics and Computers in Simulation*, February 2007. In press, also available online at http://dx.doi.org/10.1016/j.matcom.2006.12.008.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[45] D. A. JEFFERSON. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7:404–425, 1985.

[46] CLEMENS KADOW. Adaptive dynamic projection-based partitioning for parallel Delaunay mesh generation algorithms. In *SIAM Workshop on Combinatorial Scientific Computing*, San-Francisco, CA, February 2004.

[47] CLEMENS KADOW. *Parallel Delaunay Refinement Mesh Generation.* PhD thesis, Carnegie Mellon University, 2004.

[48] CLEMENS KADOW AND NOEL WALKINGTON. Design of a projection-based parallel Delaunay mesh generation and refinement algorithm. In *4th Symposium on Trends in Unstructured Mesh Generation*, Albuquerque, NM, July 2003. http://www.andrew.cmu.edu/user/sowen/usnccm03/agenda.html.

[49] G.E. KARNIADAKIS AND S.A. ORSZAG. Nodes, modes, and flow codes. *Physics Today*, 46:34–42, 1993.

[50] GEORGE KARYPIS AND VIPIN KUMAR. *MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0.* University of Minnesota, September 1998.

[51] ANDRIY KOT, ANDREY CHERNIKOV, AND NIKOS CHRISOCHOIDES. Parallel out-of-core constrained Delaunay mesh generation. In *Proceedings of the 3rd IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 183–190, Sofia, Bulgaria, September 2005.

[52] ANDRIY KOT, ANDREY CHERNIKOV, AND NIKOS CHRISOCHOIDES. Effective out-of-core parallel Delaunay mesh refinement using off-the-shelf software. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, April 2006. Available online at http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1639361.

[53] ANDRIY KOT, ANDREY N. CHERNIKOV, AND NIKOS P. CHRISOCHOIDES. Out-of-core parallel Delaunay mesh generation. In *17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation*, Paris, France, 2005. Paper T1-R-00-0710.

[54] VIPIN KUMAR, ANANTH GRAMA, ANSHUL GUPTA, AND GEORGE KARYPIS. *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms.* The Benjamin/Cummings Publishing Company, Inc., 1994.

[55] FRANCOIS LABELLE AND JONATHAN RICHARD SHEWCHUK. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the 19th ACM Symposium on Computational geometry*, pages 191–200, San Diego, CA, 2003.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[56] XIANG-YANG LI. Generating well-shaped d-dimensional Delaunay meshes. *Theoretical Computer Science*, 296(1):145–165, 2003.

[57] XIANG-YANG LI AND SHANG-HUA TENG. Generating well-shaped Delaunay meshes in 3D. In *Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37, Washington, D.C., 2001.

[58] LEONIDAS LINARDAKIS. Delaunay decoupling method for planar parallel guaranteed quality mesh generation. Master's thesis, College of William and Mary, 2003.

[59] LEONIDAS LINARDAKIS AND NIKOS CHRISOCHOIDES. Delaunay decoupling method for parallel guaranteed quality planar mesh refinement. *SIAM Journal on Scientific Computing*, 27(4):1394–1423, 2006.

[60] LEONIDAS LINARDAKIS AND NIKOS CHRISOCHOIDES. A static medial axis domain decomposition for 2D geometries. *ACM Transactions on Mathematical Software*, 2007. In press.

[61] REINALD LÖHNER AND JUAN R. CEBRAL. Parallel advancing front grid generation. In *Proceedings of the 8th International Meshing Roundtable*, pages 67–74, South Lake Tahoe, CA, 1999.

[62] GARY L. MILLER. A time efficient Delaunay refinement algorithm. In *Proceedings of the 15th annual ACM-SIAM symposium on Discrete algorithms*, pages 400–409, New Orleans, LA, 2004.

[63] GARY L. MILLER, DAFNA TALMOR, SHANG-HUA TENG, AND NOEL 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*, pages 683–692, Las Vegas, NV, May 1995.

[64] GARY L. MILLER, DAFNA TALMOR, SHANG-HUA TENG, NOEL WALKINGTON, AND HAN WANG. Control volume meshes using sphere packing: Generation, refinement and coarsening. In *Proceedings of the 5th International Meshing Roundtable*, pages 47–61, Pittsburgh, PA, October 1996.

[65] SCOTT A. MITCHELL. Cardinality bounds for triangulations with bounded minimum angle. In *Proceedings of the 6th Canadian Conference on Computational Geometry*, pages 326–331, Saskatoon, Saskatchewan, Canada, August 1994.

[66] DÉMIAN NAVE, NIKOS CHRISOCHOIDES, AND L. PAUL CHEW. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. *Computational Geometry: Theory and Applications*, 28:191–215, 2004.

[67] R. A. NICOLAIDES. Direct discretization of planar div-curl problems. *SIAM Journal on Numerical Analysis*, 29(1):32–56, 1992.

[68] T. OKUSANYA AND J. PERAIRE. Parallel unstructured mesh generation. In *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Starkville, MS, 1996.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[69] I. Pivkin, E. Hueso, R. Weinstein, D. Laidlaw, S. Swartz, and G. Karniadakis. Simulation and visualization of air flow around bat wings during flight. In *Proceedings of the International Conference on Computational Science*, pages 689–694, Atlanta, GA, 2005.

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

[71] Scott Schneider, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. Scalable locality-conscious multithreaded memor allocation. In *Proceedings of the 2006 ACM SIGPLAN International Symposium on Memory Management*, Ottawa, Canada, June 2006.

[72] Evan Conway Sherbrooke. *3-D shape interrogation by medial axial transform.* PhD thesis, Massachusetts Institute of Technology, 1995.

[73] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Ming C. Lin and Dinesh Manocha, editors, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[74] Jonathan Richard Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363, October 1997.

[75] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation.* PhD thesis, Carnegie Mellon University, 1997.

[76] Jonathan Richard Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th ACM Symposium on Computational Geometry*, pages 86–95, Minneapolis, MN, 1998.

[77] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74, May 2002.

[78] H. Si. On refinement of constrained Delaunay tetrahedralizations. In *Proceedings of the 15th International Meshing Roundtable*, pages 509–528, Birmingham, AL, September 2006. Springer.

[79] H. Si and K. Gaertner. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, San Diego, CA, September 2005. Springer.

[80] Hang Si. Tetgen version 1.4.1. http://tetgen.berlios.de/. Accessed on Aug. 3, 2006.

[81] Daniel A. Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement: Algorithms and analyses. In *Proceedings of the 11th International Meshing Roundtable*, pages 205–217, Ithaca, NY, 2001.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[82] DANIEL A. SPIELMAN, SHANG-HUA TENG, AND ALPER ÜNGÖR. Time complexity of practical parallel Steiner point insertion algorithms. In *Proceedings of the 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 267–268, Barcelona, Spain, 2004. ACM Press.

[83] SHANG-HUA TENG, February 2004. Personal Communication, February 2004.

[84] TetMesh-GHS3D V3.1 The fast, reliable, high quality tetrahedral mesh generator and optimiser. White paper. http://www.simulog.fr/mesh/tetmesh3p1d-wp.pdf. Accessed on Feb. 27, 2004.

[85] TIANKAI TU, DAVID R. O'HALLARON, AND OMAR GHATTAS. Scalable parallel octree meshing for terascale applications. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, Seattle, WA, 2005. IEEE Computer Society.

[86] ALPER ÜNGÖR. Off-centers: A new type of Steiner points for computing size-optimal guaranteed-quality Delaunay triangulations. In *Proceedings of LATIN*, pages 152–161, Buenos Aires, Argentina, April 2004.

[87] ALPER ÜNGÖR. Quality triangulations made smaller. In *Proceedings of the European Workshop on Computational Geometry*, Technische Universiteit Eindhoven, March 2005. Accessed online at http://www.win.tue.nl/EWCG2005/Proceedings/2.pdf on April 28, 2007.

[88] ROY A. WALTERS. Coastal ocean models: Two useful finite element methods. *Recent Developments in Physical Oceanographic Modeling: Part II*, 25:775–793, 2005.

[89] TIMOTHY WARBURTON. *Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications.* PhD thesis, Brown University, 1999.

[90] DAVID F. WATSON. Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes. *Computer Journal*, 24:167–172, 1981.

[91] ROGER WEBSTER. *Convexity.* Oxford Science Publications, 1994. Page 200.

[92] GEORGE ZAGARIS, SHAHYAR PIRZADEH, ANDREY CHERNIKOV, AND NIKOS CHRISOCHOIDES. Parallel mesh generation for CFD simulations of complex real-world aerodynamic problems. In *6th Symposium on Trends in Unstructured Mesh Generation*, San Francisco, CA, July 2007. To appear.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

VITA

Andrey Nikolayevich Chernikov

Andrey Nikolayevich Chernikov was born in Donetsk, Ukraine in 1978. In 1995 he finished High School 18 in Nalchik, Russia with a Golden Medal award. He entered the Department of Mathematics of the Kabardino-Balkar State University in Nalchik, where he received his Bachelor's (in 1999) and Master's (in 2001) degrees, both with Major in Applied Mathematics and Computer Science, and both with the highest distinction (red) diploma. His Bachelor's thesis was on the methodology and the development of software tools for teaching introductory Computer Science courses. His Master's thesis was on the modeling of ionic systems by the Molecular Dynamics method. From 1997 to 2001 he also worked on the mathematical and software development problems of building geographic information systems at the Institute of Informatics and Problems of Regional Management of Russian Academy of Sciences. In Fall 2001 he began studying toward a Ph.D. degree in the Department of Computer Science at the College of William and Mary, where he is currently a Ph.D. candidate and a research assistant. From June to December 2006 he worked as a visiting research associate in the Division of Applied Mathematics at Brown University.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.