Overview of Parallel Mesh Generation and Optimization Methods

Andrey Chernikov[1], Suzanne Shontz[2], and Nikos Chrisochoides[1]

[1]Department of Computer Science
Center for Real-Time Computing
Old Dominion University
achernik@cs.odu.edu, nikos@cs.odu.edu

[2]Department of Mathematics and Statistics
Center for Computationsl Sciences
Department of Computer Science and Engineering
Computational Engineering Graduate Program
Mississippi State University
sshontz@math.msstate.edu

February 20, 2014

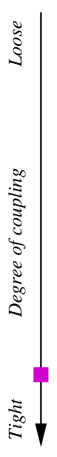
Outline

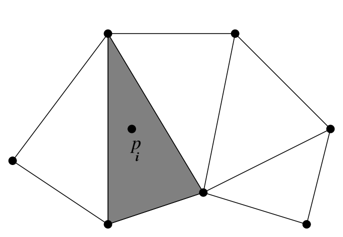
1. A Taxonomy of Parallel Delaunay Meshing Algorithms
2. Other Parallel Meshing Algorithms
3. Parallel Mesh Optimization Algorithms

Outline

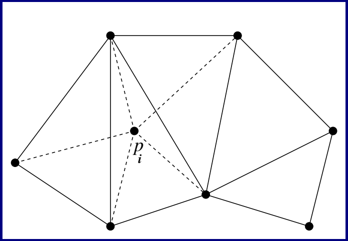
1. A Taxonomy of Parallel Delaunay Meshing Algorithms
2. Other Parallel Meshing Algorithms
3. Parallel Mesh Optimization Algorithms

Optimistic Delaunay Meshing

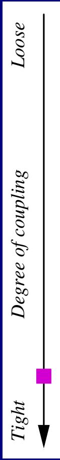


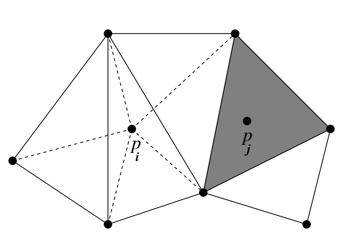


Optimistic Delaunay Meshing

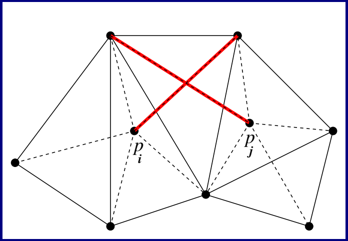


Optimistic Delaunay Meshing

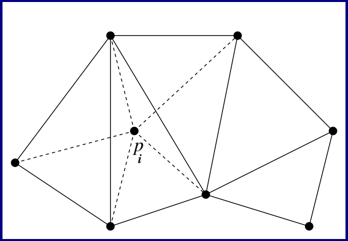




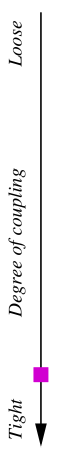
Optimistic Delaunay Meshing

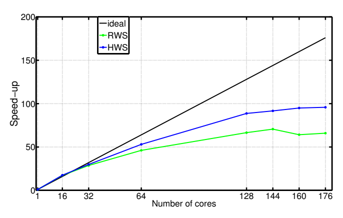


Optimistic Delaunay Meshing

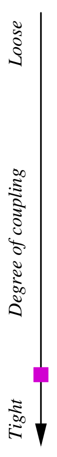


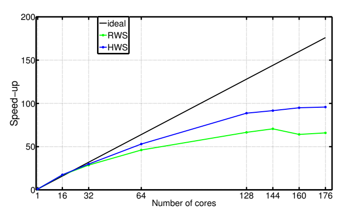
Optimistic Delaunay Meshing

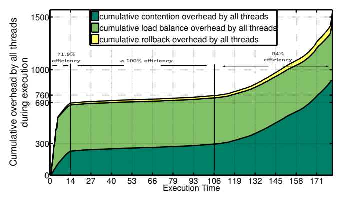




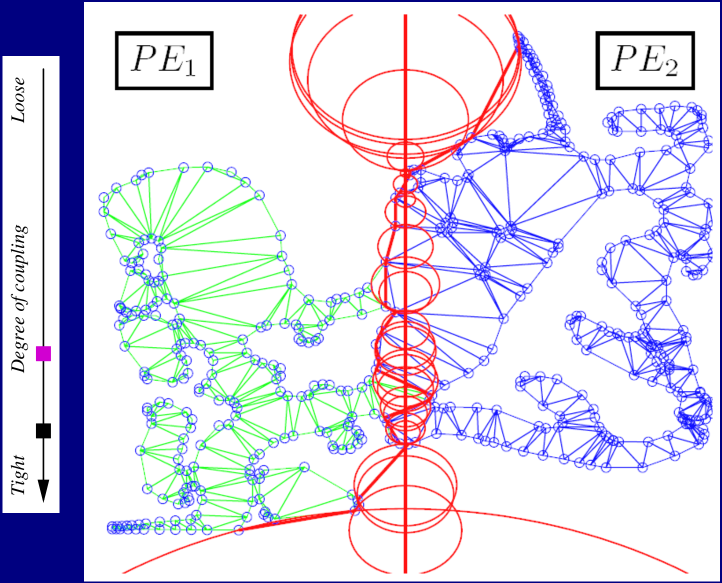
Optimistic Delaunay Meshing



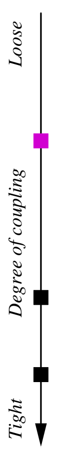


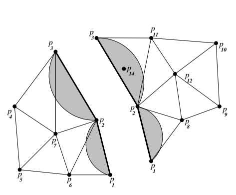


Parallel Projection-Based Delaunay Meshing

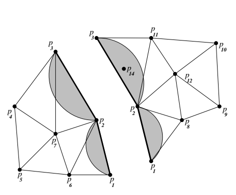


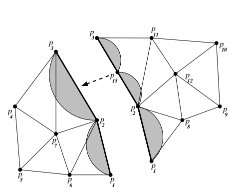
Parallel Constrained Delaunay Meshing



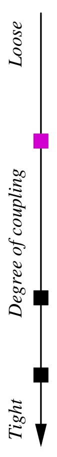


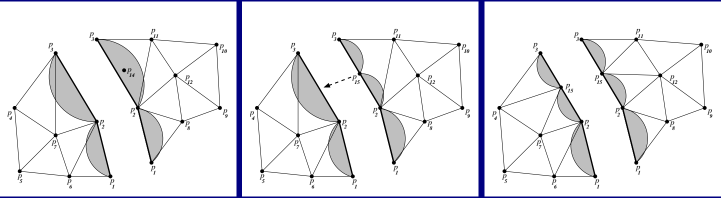
Parallel Constrained Delaunay Meshing



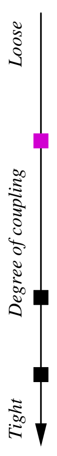


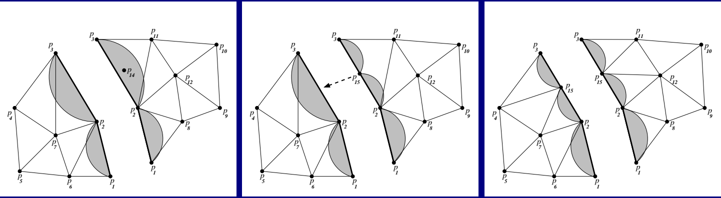
Parallel Constrained Delaunay Meshing

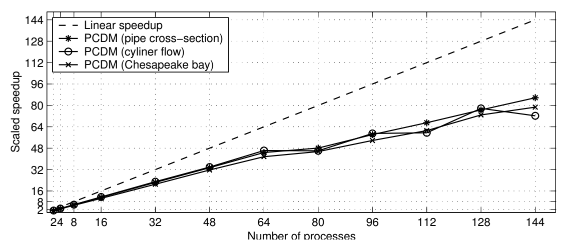




Parallel Constrained Delaunay Meshing

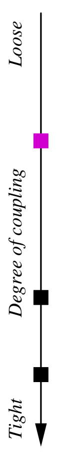


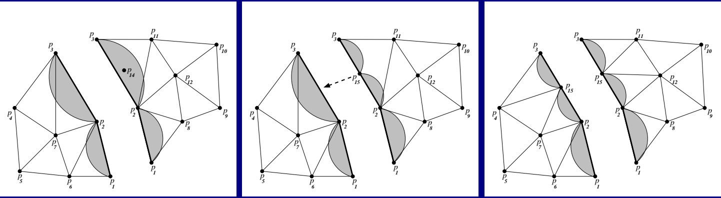


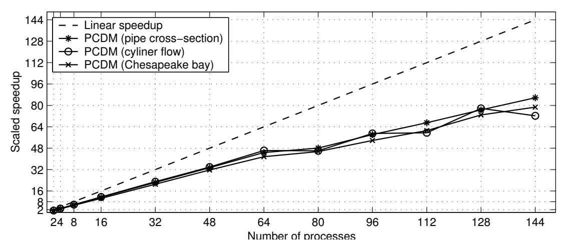


Scaled speedup: the number of triangles $\approx 10M \times P$, that is, for 2 processors $20M$, and for 144 processors about $1.4B$.

Parallel Constrained Delaunay Meshing



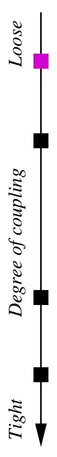




Scaled speedup: the number of triangles $\approx 10M \times P$, that is, for 2 processors $20M$, and for 144 processors about $1.4B$.

(Extension to 3D subject to availability of a 3D domain decomposer)

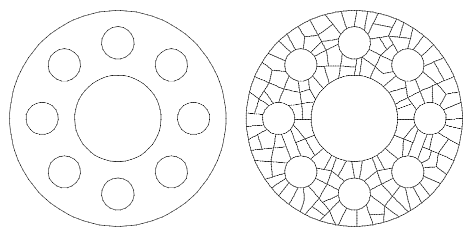
Domain Decomposition and Decoupling



Given 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:

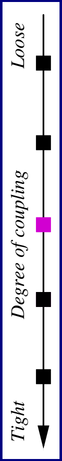
$$\Omega = \bigcup_{i=1}^{N} \Omega_i, \quad \partial\Omega_i \cap \partial\Omega_j = S_{ij}, \quad i, j = 1, \ldots, N, \quad i \neq j,$$

while the separators do not create very small angles and other features.



(Has not been extended to 3D)

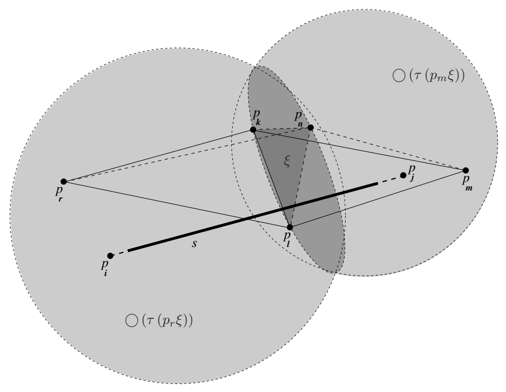
Parallel Delaunay Refinement Method



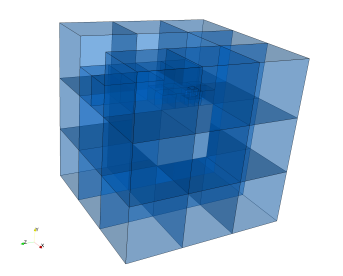
- No rollbacks
- No fine-grain synchronization
- Does not require to solve the domain decomposition problem
- Extended to 3D
- Code reuse

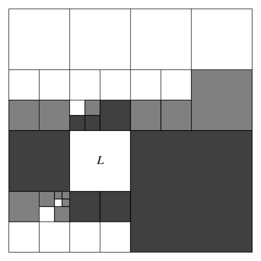
Sufficient Condition for Graded PDR (3D)

Lemma (*Sufficient condition of Delaunay-independence*) Points p_i and p_j are Delaunay-independent if there exists a subsegment $s \subseteq \mathcal{L}(p_i p_j)$ such that $\forall \tau \in T \; : \; s \cap \bigcirc(\tau) \implies 2r(\tau) \leq |s|$.



Buffer Zone (3D)





Definition ***3D buffer zone*** is the set of leaves

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

under the condition

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

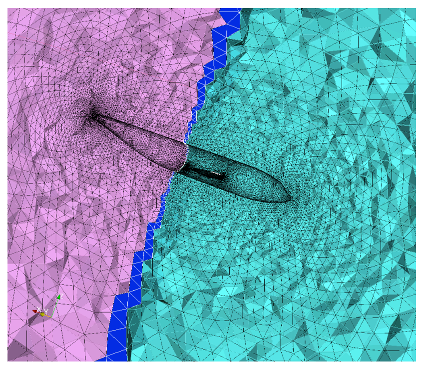
Overview

Parallel Delaunay Meshing Methods / ***Properties***	Stability	No domain decomposition	No fine grain synchronization	No rollbacks	Extended to 3D	Code reuse
Parallel Optimistic Delaunay Meshing *[Nave, Chrisochoides, and Chew, 2003, 2004]*	●	●			●	
Parallel Constrained Delaunay Meshing *[Chernikov and Chrisochoides, 2006]*	●		●	●		
Parallel Projection–Based Delaunay Meshing *[Kadow and Walkington, 2004]*	●	●				
Parallel Domain Delaunay Decoupling *[Linardakis and Chrisochoides, 2006, 2008]*	●		●	●		●
Parallel Generalized Delaunay Refinement *[Chernikov and Chrisochoides, 2005–2009]*	●	●	●	●	●	●

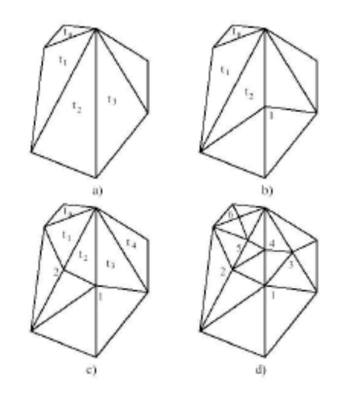
Outline

Parallel Advancing Front Meshing

- ***Idea:*** Given a final surface mesh of domain *D* construct a 3D zone using a pre-computed surface *S* to guide a single layer along *S* starting from any external boundary of *D*.
- Given a source driven AFT, a zone can be constructed from elements whose size will remain invariant throughout the mesh generation process.
- No new features or small angles due to decomposition, therefore any decomposition works.
- Caveat: termination not guaranteed for the sub-problems.

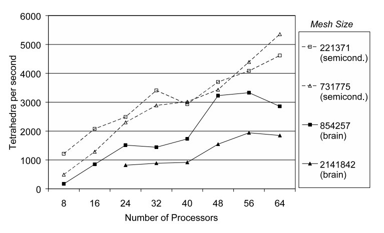


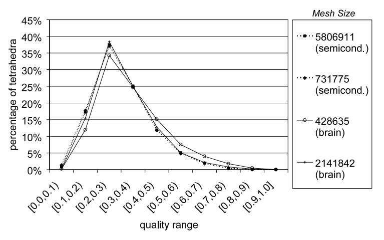
Parallel Terminal Edge Bisection



- A ***terminal-edge*** is the longest edge of every element that shares such an edge.
- A ***terminal star*** is the set of elements that share a terminal-edge.
- The stopping criteriob is the predefined bound for the length of the terminal edges.
- The terminal-star algorithm eliminates the management of non-conforming edges both in the interior of the submeshes and in the interfaces i.e., <u>eliminates communication</u>.

Parallel Terminal Edge Bisection





Qulaity is measured as normalized volume / (longest edge)^3.

Outline

1. A Taxonomy of Parallel Delaunay Meshing Algorithms
2. Other Parallel Meshing Algorithms
3. Parallel Mesh Optimization Algorithms

An Overview of Parallel Triangular/Tetrahedral Mesh Optimization Methods

There are **two types** of **parallel mesh optimization methods**:

- mesh smoothing methods
- mesh untangling methods.

Current literature: 4 parallel mesh smoothing algorithms; 1 parallel mesh untangling algorithm; 1 combined algorithm.

Additional methods: 1 parallel mesh smoothing algorithm; 1 parallel mesh untangling method. Under review in paper by Sastry and Shontz.

Totals: 5 parallel mesh smoothing algorithms; 2 parallel mesh untangling method; 1 combined algorithm.

Parallel Mesh Smoothing Methods

First parallel mesh smoothing algorithm: Proposed by Freitag, Jones, and Plassmann in SISC in 1999.

Algorithm and architecture: parallel nonsmooth optimization algorithm for tetrahedral meshes on distributed memory machines.

Graph coloring: used to identify independent sets of vertices.

For each independent set:

1. Optimize the locations of vertices of color i using local vertex movement.
2. Communicate the new vertex positions.

Communication/synchronization: Root process performs unstructured, asynchronous communication.

Freitag/Jones/Plassmann Method (continued):

Synchronization: No global synchronization is needed, as only one independent set of vertices is smoothed at a time.

Algorithmic Results: Parallel efficiencies of up to $70 - 88\%$ were obtained for up to 64 processors when run on the IBM SP. Efficiencies were not reported for their runs on the ATM connected SPARC Ultras.

Theoretical Results: For a Parallel Random Access Machine (PRAM) version of the algorithm: Provably fast runtime bound, correct execution.

Parallel Mesh Smoothing Methods

Parallel feature-preserving mesh smoothing algorithm: Proposed by Jiao and Alexander at 2005 ICCSA Conference.

Algorithm and architecture: Parallel feature-preserving triangular surface mesh smoothing algorithm on distributed memory machines

Key concept: Medial quadric (extension of quadric involving medial axis). Used in feature detection.

Approach: Detect features, such as edges, corners, cusps, and one-sided normals along edges. Move vertices while preserving the shape and features of a surface as follows. Ridge vertices are moved before smooth vertices.

Results: 43% of maximum parallel efficiency when implented on distributed memory computers with up to 128 processors.

Parallel Mesh Smoothing Methods

Parallel anisotropic mesh smoothing algorithm: Proposed by Gorman, Southern, Farrell, Piggott, Rokos, and Kelly at 2012 ICCS Conference.

Algorithm and architecture: Hybrid OpenMP/MPI anisotropic mesh smoothing algorithm for cache coherent nonuniform memory access (ccNUMA) machines.

Architecture: Has many cores per node. Thus, there are aspects of distributed memory (node-to-node) and shared memory (core-to-core).

Parallelism: Message passing paradigm for distributed memory (MPI); thread-based parallelism for shared memory (OpenMP). OpenMP is preferred due to greater potential for use with co-processors. Easy to extend (older) MPI methods.

Parallel Mesh Smoothing Methods

Gorman et al. algorithm (continued)

Smoothing kernels: Quality-constrained Laplacian smoothing and nonsmooth optimization.

Graph coloring: Uses parallel graph coloring algorithm to identify independent sets of vertices for use with local mesh optimization.

Data locality: Three techniques: Partitioning of Linux kernel memory (page faults); processor affinity between threads and CPUs; vertex reordering (fill-reducing).

Progressive domain masking: Only smooth vertices which need updating. Effect: reduced data locality but better load balancing.

Results: Dual-socket Intel Westmere server, with each socket consisting of a 6-core Xeon CPU X5650 @ 2.67 GHz. High degree of concurrency and fine grained scaling behavior obtained.

Parallel Untangling and Smoothing Algorithm

Parallel untangling and smoothing algorithm: Proposed by Benitez, Rodriguez, Escobar, and Montenegro at IMR 2013

Algorithm and architecture: OpenMP parallel simultaneous untangling and smoothing for tetrahedral meshes on shared-memory, many-core machines

Key concept: use a modified mesh quality metric (without singularities) which allows for simultaneous untangling and smoothing of meshses

Graph coloring: Various graph coloring algorithms are used to identify independent sets of vertices for use with local mesh optimization.

Graph coloring techniques: Luby's Monte Carlo algorithm (serial), parallel version of previous algorithm, Bozdag's parallel greedy coloring algorithm.

Parallel Untangling and Mesh Smoothing Algorithm

Montenegro et al. algorithm (continued):

Architectures for experiments: (1) HP Integrity Superdome node that contains 128 Itanium 2 Montvale cores with 1.6 GHz clock speed, 1024 GB NUMA shared memory; (2) Manycore Testing Lab: 40 Westmere 2.27 GHz cores and 252 GB NUMA shared memory.

Results: On 128 cores: Parallel efficiency: 76% and up (main optimization procedure), 50% and up (entire algorithm). **On 40 cores:** The parallel efficiency is reduced. **No winner:** There is no best graph coloring algorithm.

Observation: OpenMP loop-scheduling overhead: mainly responsible for performance deterioration and load imbalance when observed.

Parallel Untangling and Smoothing Algorithms

Parallel untangling and smoothing algorithms: Proposed by Sastry and Shontz in 2013 (under review).

Algorithm and architecture: OpenMPI parallel nonlinear mesh optimization on distributed memory architectures. Can be used for **either mesh smoothing** (with any mesh quality metric) **or mesh untangling** (with appropriate choice of metric).

Approach: Global mesh smoothing.

Graph coloring: Used to identify independent **edges** in graph of communicating processes (not mesh edges). Used to synchronize unstructured communication.

Results: Run on Intel Xeon CPU E-7-4870 cluster with 80 cores with 2.40 GHz clock speed, 750 GB RAM. **Strong scaling efficiency:** 80% on 64 cores. **Weak scaling efficiency:** good.

Conclusions

1. We reviewed existing parallel mesh smoothing and untangling methods.
2. All of the methods with the exception of the method by Sastry and Shontz involve local mesh optimization. The latter methods involve global mesh optimization.
3. The methods were developed for distributed memory or shared memory machines; the exception was the hybrid method by Gorman et al.
4. **Important concepts:** graph coloring, vertex reordering, load balancing, scheduling, etc.

Promising Research Directions

1. Simultaneous parallel mesh construction and optimization
2. Comparison study of existing methods
3. More methods are needed!
4. Hybrid methods (distributed and shared memory)
5. Architectures: GPUs, co-processors, etc.
6. Vertex reordering (for both local and global methods)
7. Mesh partitioning (local/patch/global)
8. Graph coloring
9. Load balancing, scheduling, performance modeling, etc.

Acknowledgments

- NSF CAREER Award ACI-1330056 (formerly ACI-1054459)
- NSF PECASE Award

References I

Andrey Chernikov and Nikos Chrisochoides.
Practical and efficient point insertion scheduling method for parallel guaranteed quality Delaunay refinement.
In *ACM International Conference on Supercomputing*, pages 48–57, Saint-Malo, France, June 2004.

Andrey Chernikov and Nikos Chrisochoides.
Parallel 2D graded guaranteed quality Delaunay mesh refinement.
In *International Meshing Roundtable*, pages 505–517, San Diego, CA, September 2005.

Andrey Chernikov and Nikos Chrisochoides.
Generalized Delaunay mesh refinement: from scalar to parallel.
In *International Meshing Roundtable*, pages 563–580, Birmingham, AL, September 2006a.

Andrey Chernikov and Nikos Chrisochoides.
Parallel guaranteed quality Delaunay uniform mesh refinement.
SIAM Journal on Scientific Computing, 28:1907–1926, November 2006b.

Andrey Chernikov and Nikos Chrisochoides.
Three-dimensional Delaunay refinement for multi-core processors.
In *ACM International Conference on Supercomputing*, pages 214–224, Island of Kos, Greece, June 2008a.

Andrey Chernikov and Nikos Chrisochoides.
Algorithm 872: parallel 2D constrained Delaunay mesh generation.
ACM Transactions on Mathematical Software, 34:6–25, January 2008b.

Andrey Chernikov and Nikos Chrisochoides.
A template for developing next generation parallel Delaunay refinement methods.
Finite Elements in Analysis and Design, 46:96–113, 2010.

Nikos Chrisochoides and Démian Nave.
Parallel Delaunay mesh generation kernel.
International Journal for Numerical Methods in Engineering, 58:161–176, 2003.

References II

Panagiotis Foteinos and Nikos Chrisochoides.
High quality real-time image-to-mesh conversion for finite element simulations.
In *ACM International Conference on Supercomputing*, Eugene, OR, 2013. ACM.

Panagiotis Foteinos and Nikos Chrisochoides.
High quality real-time image-to-mesh conversion for finite element simulations.
Journal on Parallel and Distributed Computing, 74(2):2123–2140, 2014.

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.

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.

Leonidas Linardakis and Nikos Chrisochoides.
Graded Delaunay decoupling method for parallel guaranteed quality planar mesh generation.
SIAM Journal on Scientific Computing, 30(4):1875–1891, March 2008a.

Leonidas Linardakis and Nikos Chrisochoides.
Algorithm 870: A static geometric medial axis domain decomposition in 2D Euclidean space.
ACM Transactions on Mathematical Software, 34(1):1–28, 2008b.

Démian Nave, Nikos Chrisochoides, and L. Paul Chew.
Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains.
In *Proceedings of the 18th ACM Symposium on Computational Geometry*, pages 135–144, Barcelona, Spain, 2002. ISBN 1-58113-504-1.

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.

References III

Maria-Cecilia Rivara, Daniel Pizarro, and Nikos Chrisochoides.
Parallel refinement of tetrahedral meshes using terminal-edge bisection algorithm.
In *13th International Meshing Roundtable*, pages 427–436, Williamsburg, VA, September 2004.

George Zagaris, Shahyar Pirzadeh, and Nikos Chrisochoides.
A framework for parallel unstructured grid generation for practical aerodynamic simulations.
In *47th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2009.