Sealing Faceted Surfaces to Achieve Watertight CAD Models

Brandon M. Smith[1], Timothy J. Tautges[2], Paul P.H. Wilson[1]

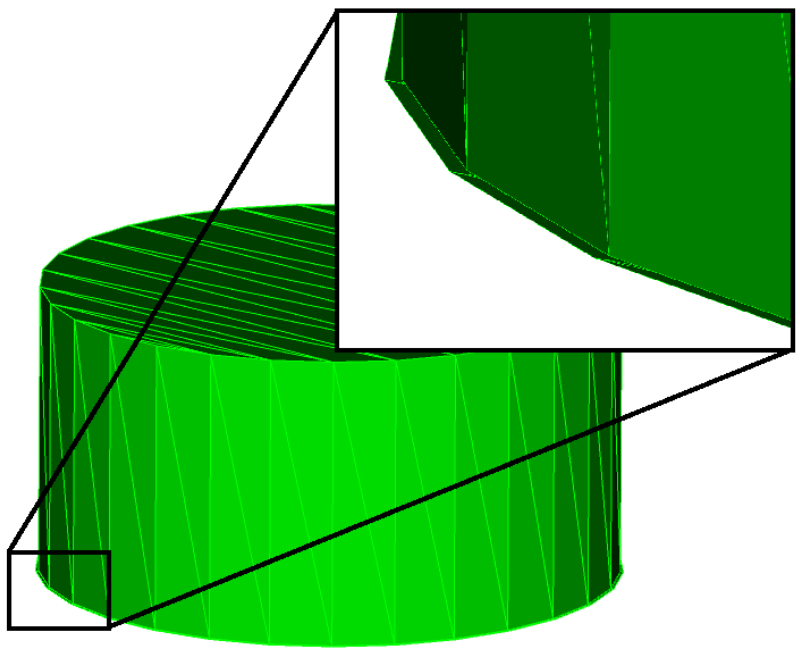
[1]University of Wisconsin-Madison
[2]Argonne National Laboratory

October 4, 2010

19th International Meshing Roundtable

Facet Based Models (FBMs)

- Geometric Vertices, Edges, and Faces represented by points, edges, and triangles
- Used for file transfer, mesh generation, and efficient geometric computation
- Solid modeling engines typically facet each Face independently
- Faceted boundaries of neighboring Faces are not the same
- Geometric gaps and discontinuous topology occur between Faces
- Problematic for meshing, ray tracing, etc...



Problem Statement: Create algorithm to fix faceting flaws between Faces.

Two Approaches to Fix Faceting Flaws

Attempt to restore missing or discarded information.

Region-Based

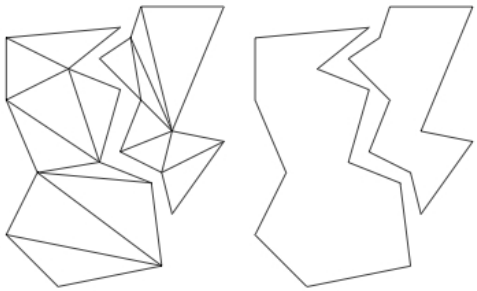
- Reconstruct entire boundary using intermediate representation
- Often theoretical guarantee
- α-Shapes [Edelsbrunner and Mücke, 1994],
  Crust [Amenta et al., 1998],
  Binary Space Partition [Murali and Funkhouser, 1997],
  Marching Cubes [Ju, 2004]

Mesh-Based

- Quickly locate defects using free edges
- Repair flaws with small perturbations to boundary in vicinity of defects
- Preserve Face features away from local defects
- Rely on proximity

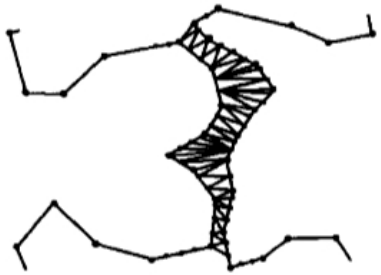
Mesh-Based Approach Chosen

Mesh-Based Approaches



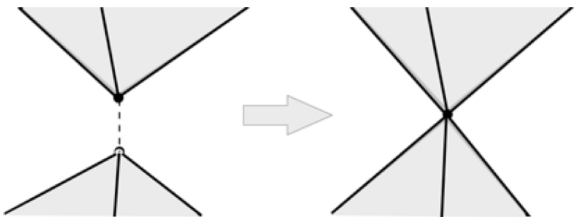
Assemble Free Edges [Barequet and Kumar, 1997]

1. Free edges adjacent to a single facet
2. Connect to form arcs using topology
3. Associate arcs by proximity or topology
4. Possible ambiguity due to pinch points



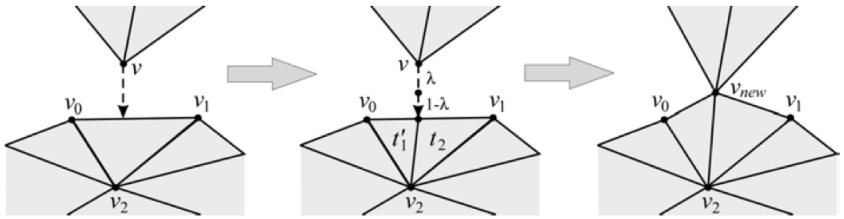
Triangulate [Barequet and Sharir, 1995]

1. Create new triangles between assembled free edges
2. Use Ear clipping or stitching
3. Locally optimize patch by angle or heuristic



Vertex-Vertex Contraction

- Tolerance vs. feature size
- Location (one,average,bucket)
- Avoids new triangles
- Edge proximity $\neq$ vertex proximity



Vertex-Edge Contraction [Borodin et al., 2002]

1. Project vertex onto edge
2. Insert vertex into the edge at projection
3. Split triangle into two triangles
4. Perform vertex-vertex contraction

Algorithm Requirements

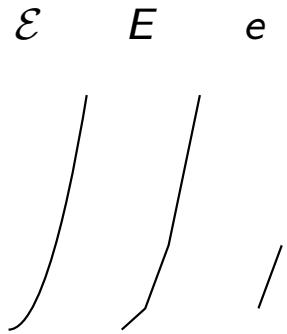
1. Seal faceted Faces along Edges to create a watertight model.
2. To preserve human efficiency, the algorithm must be automatic.
3. New facets must be owned by exactly one Face.
4. Support non-manifold Faces.
5. Fast enough to use as a preprocessing module.
6. Deformation of input model should be minimized.
7. Creation of new triangles should be minimized.

Contribution: Increase robustness by using topology of Edges to develop a provably reliable algorithm implemented as open-source software.

Assumptions I

Notation

- Geometric Entities: vertices $\mathcal{V}$, edges $\mathcal{E}$, faces $\mathcal{F}$, regions $\mathcal{R}$, loops $\mathcal{L}$
- Faceted Entities: vertices V, edges E, faces F, regions R, loops L
- Faceted Elements: facet edges e, facet faces f, facet points p
- Distance $d()$, boundary $\Omega()$, facet tolerance ε_f, merge tolerance ε_m



Geometry Assumptions

1. The geometry is a cell complex: $\{\mathcal{V}^i, \mathcal{E}^i, \mathcal{F}^i, \mathcal{R}^i\} \in C$

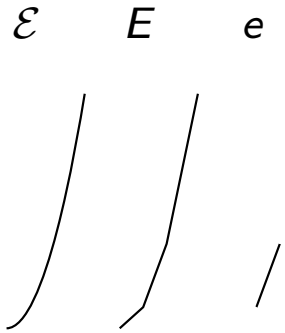
Faceting Assumptions

2. Each geometric Edge and Face has a corresponding faceted entity: $\mathcal{E}^i \longleftrightarrow E^i$, $\mathcal{F}^i \longleftrightarrow F^i$, $\forall i$.
3. Individual faceted Edges and Faces are a cell complex: $E^i \in C, F^i \in C$, $\forall i$.
4. Feature size is much larger than facet tolerance: $LFS \gg \varepsilon_m$.
5. Each faceting is non-degenerate, oriented, and non-inverted.
6. Faceting for each Edge with a single Vertex will have at least 3 facet edges.

Assumptions II

Notation

- Geometric Entities: vertices $\mathcal{V}$, edges $\mathcal{E}$, faces $\mathcal{F}$, regions $\mathcal{R}$, loops $\mathcal{L}$
- Faceted Entities: vertices V, edges E, faces F, regions R, loops L
- Faceted Elements: facet edges e, facet faces f, facet points p
- Distance $d()$, boundary $\Omega()$, facet tolerance ε_f, merge tolerance ε_m



Connecting Assumptions

7. Each faceted Face boundary corresponds to a set of faceted Edges: $\Omega(\{f_j\})^i \rightarrow \{E^k\}\ \forall i,j,k.$
8. Facet points are within merge tolerance of corresponding geometric entities: $d(p^i,\mathcal{E}^j) \leq \varepsilon_f,\ d(p^k,\mathcal{F}^l) \leq \varepsilon_f\ \forall i,j,k,l.$
9. Facet edges and triangles are within facet tolerance of corresponding geometric entities: $d(E^i,\mathcal{E}^i) \leq \varepsilon_f,\ d(F^j,\mathcal{F}^j) \leq \varepsilon_f\ \forall i,j.$
10. The facet tolerance is much greater than the merge tolerance: $\varepsilon_f \gg \varepsilon_m.$
11. Points on faceted face boundaries are within merge tolerance of *some* model entity that bounds the corresponding model entity, though *which* bounding model entity is not known: $d(\Omega(\{f_j\})^i,\mathcal{E}^k) \leq \varepsilon_f\ \forall i,j,k.$
12. The elements of the faceted edges are not the same as the boundary of the elements of the faceted faces: $\{e_j\}^i \notin \Omega(\{f_l\}^k)\ \forall i,j,k,l.$

Algorithm Overview

Import FBM from solid modeling engine

1. Preprocess small entities
2. Seal Faces
   - Seal arcs to corresponding Edges
3. Postprocess inverted triangles

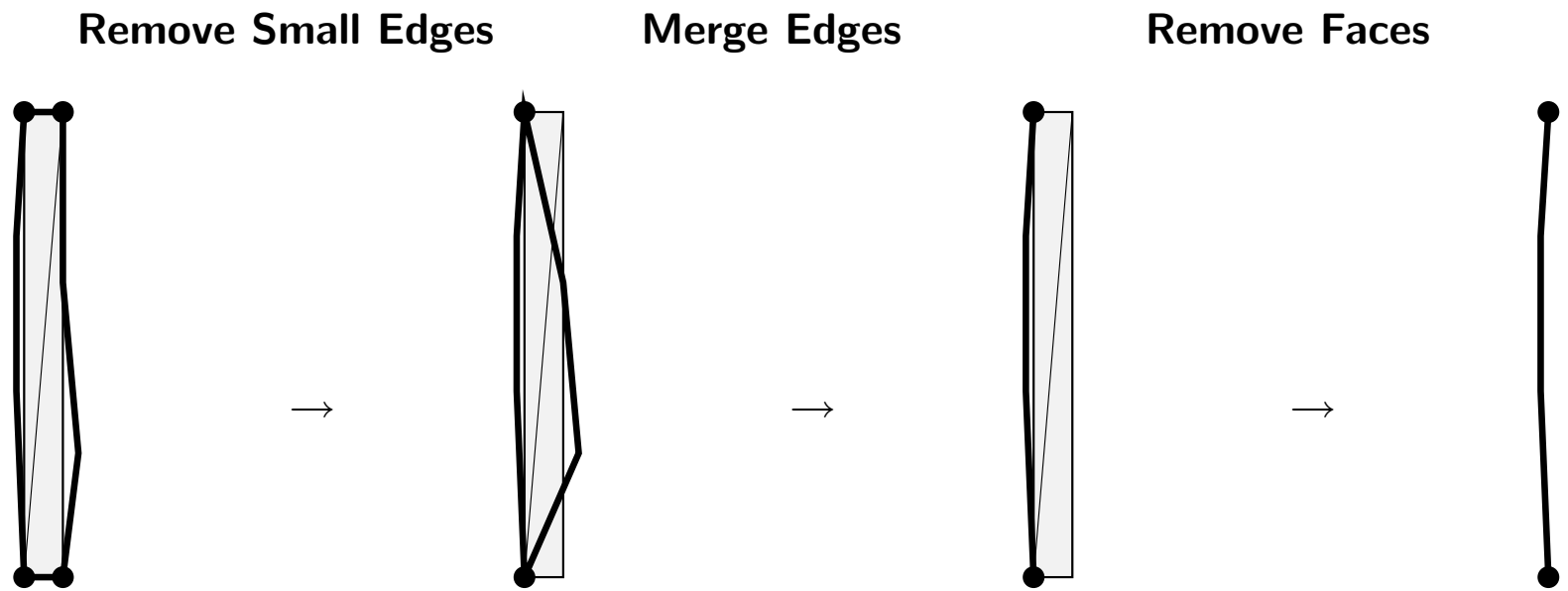
Export watertight FBM to application/library

- Algorithm operates only on FBM, not solid model.
- *Outline of proof appears in paper.*

Preprocess Small Entities

1. Remove Edges shorter than facet tolerance
2. Merge Edges that average less than facet tolerance apart from each other
3. Remove Faces if all bounding Edges occur in merged pairs
4. Remove Regions if all Faces have been removed

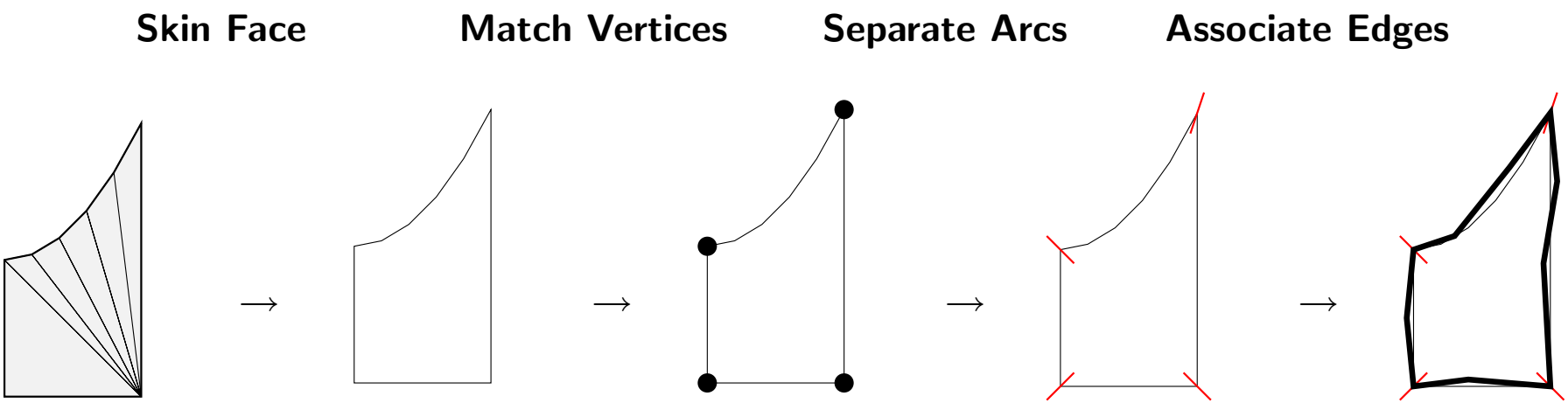
Small entities often created during imprinting.



Seal Faces

1. Skin Face to recover bounding edges
2. Orient bounding edges
3. Assembled bounding edges into loops
4. Match Vertex points to points on loops, using proximity
5. Separate bounding loops into arcs, using Vertex points as separators
6. Associate arcs with corresponding Edges that bound the Face
7. If Edge has not yet been sealed, replace Edge with arc
8. Otherwise seal_arcs to corresponding Edges that bound the Face

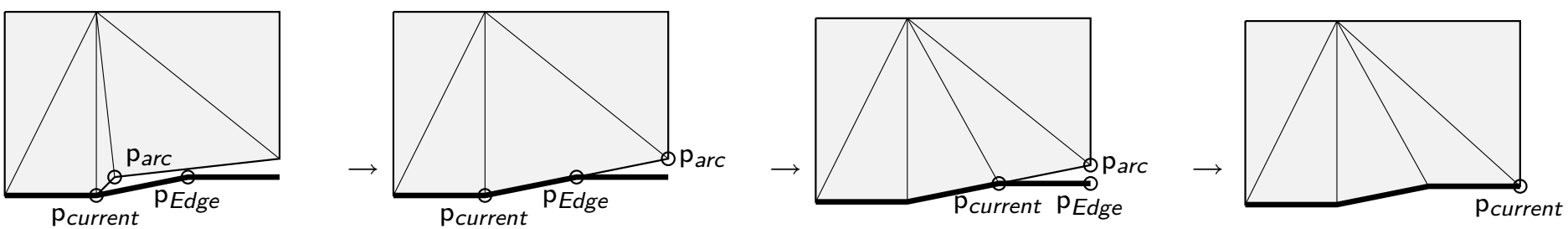
See paper for algorithm.



Seal Arcs

Initialize $p_{current}$, p_{Edge}, and p_{arc}
Until arc is sealed:
- $p_{next} = p_{Edge}$ or p_{arc} s.t. $d(p_{current}, p_{next}) = \min$
- If $d(p_{current}, p_{next}) \leq \varepsilon_f$ — Vertex-Vertex Contraction
  - contract p_{next} to $p_{current}$
- Else if $d(p_{Edge}, p_{arc}) \leq \varepsilon_f$ — Vertex-Vertex Contraction
  - contract p_{arc} to p_{Edge}
- Else — Vertex-Edge Contraction
  - insert p_{next} into opposite edge
- Update adjacencies, remove degeneracies
- Update $p_{current}$, p_{Edge}, and p_{arc} as needed — *See paper for algorithm.*

Vertex-Vertex Contraction (to $p_{current}$) **Vertex-Edge Contraction** **Vertex-Vertex Contraction** (to p_{Edge})



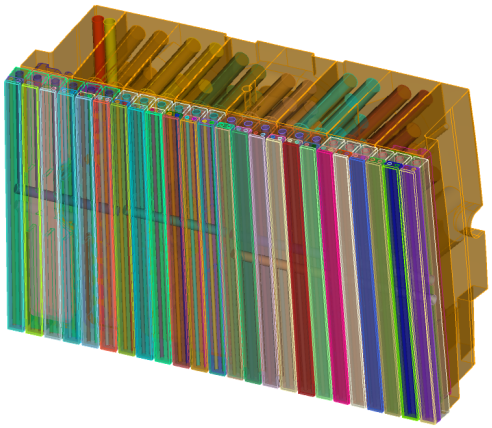
Post Process Inverted Triangles

1. Remove inverted triangles
2. Refacet loops
3. Expand patches as necessary
4. Constrain to bounding Edges

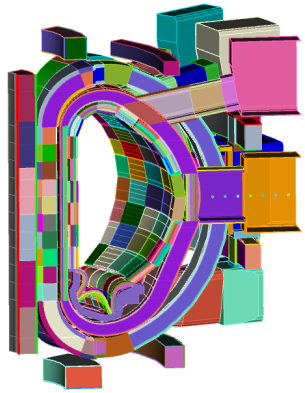
Inverted facets occur when assumptions are not true:

$\varepsilon_f > LFS$ or $\varepsilon_f < \varepsilon_m$.

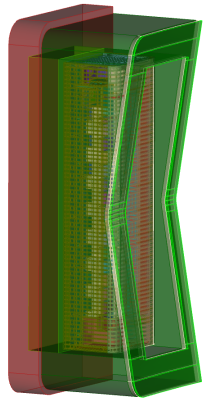
Test Models



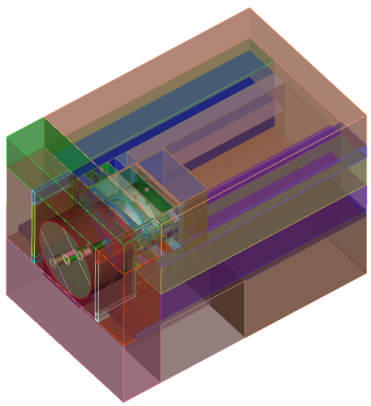
ITER FWS Module 13



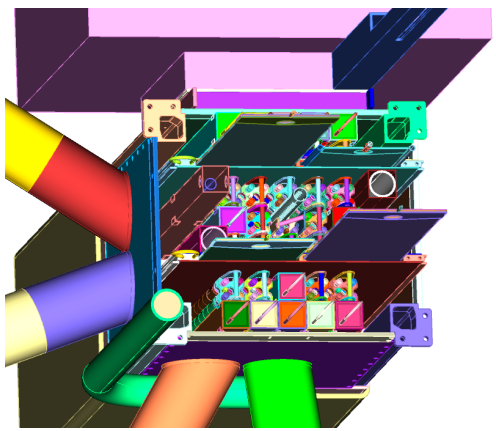
40° ITER Benchmark



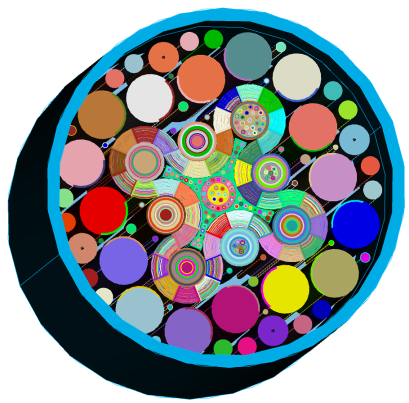
ITER Test Blanket Module



FNG Benchmark



UW Nuclear Reactor



Advanced Test Reactor

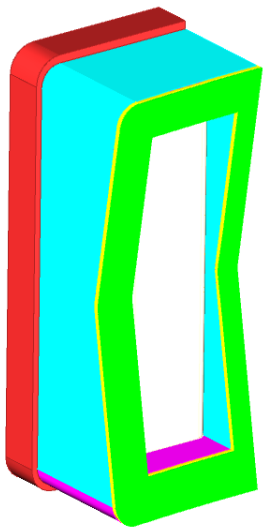
Entity Count

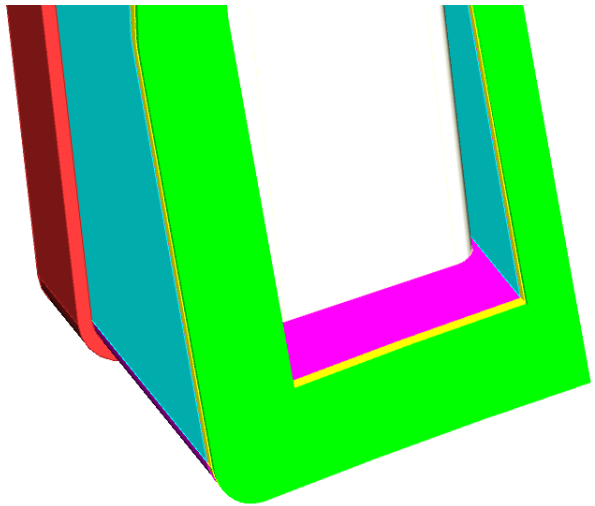
Table: Geometric entity count and number of triangular facets [millions] as a function of facet tolerance [cm].

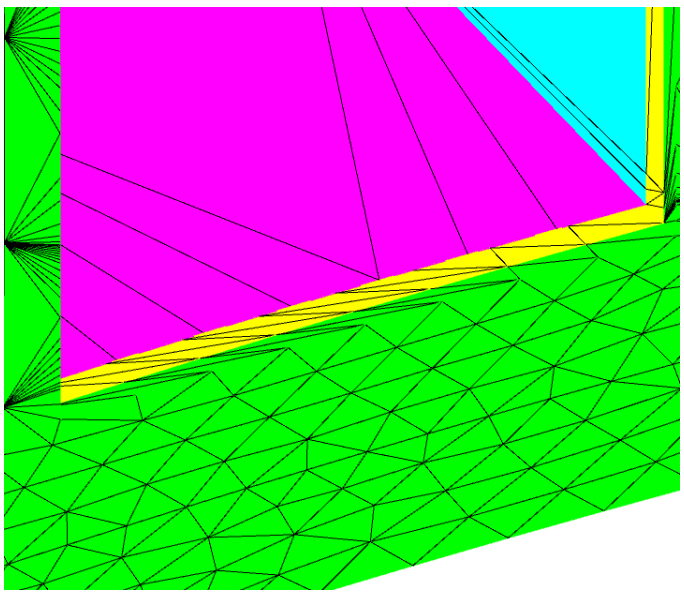
Model	Geometric Entity			Facet Tolerance [cm]				
	Regions	Faces	Edges	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
UW Nuclear Reactor	2820	30237	65078	2.62	2.62	2.98	8.56	29.1
Advanced Test Reactor	2132	11827	22402	0.44	0.45	0.84	2.44	7.65
40° ITER Benchmark	902	9834	20485	0.32	0.78	2.07	8.76	16.3
ITER Test Blanket Module	71	4870	13625	0.07	0.08	0.12	0.38	1.57
ITER Module 4	155	4155	10255	0.29	0.29	0.34	1.07	2.89
ITER Module 13	146	2407	5553	0.28	0.29	0.50	2.54	8.65
FNG Fusion Benchmark	1162	4291	5134	0.11	0.11	0.14	0.46	1.14
ARIES First Wall	3	358	743	0.17	0.87	1.21	1.55	2.45
High Average Power Laser	15	139	272	0.15	0.47	0.53	0.61	0.88
Z-Pinch Fusion Reactor	24	95	143	0.05	0.29	0.99	1.17	1.53

Defaults: $\varepsilon_f = 10^{-3}$ cm, $\varepsilon_m = 5x10^{-4}$ cm

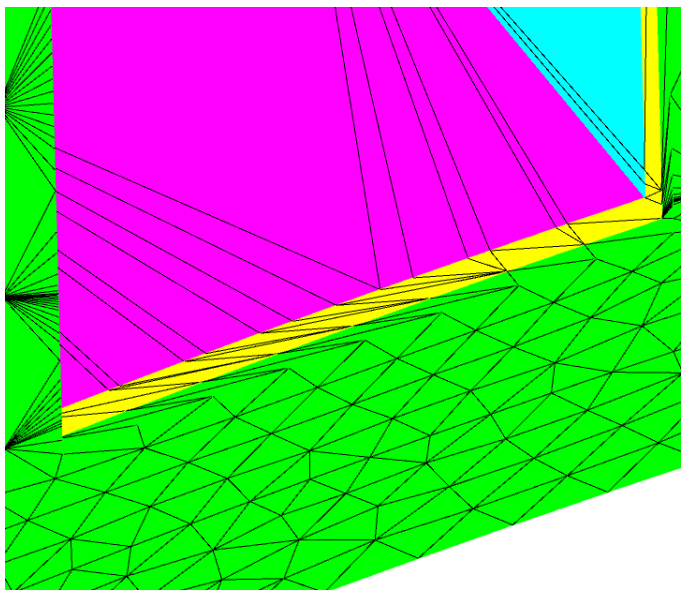
Example: ITER Test Blanket Module







Unsealed



Sealed

Face Sealing Failures

Table: Number of face sealing failures as a function of facet tolerance [cm].

Model	Facet Tolerance [cm]				
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
UW Nuclear Reactor	1019	0	0	0	0
Advanced Test Reactor	88	0	0	0	0
40° ITER Benchmark	18	9	0	18	191
ITER Test Blanket Module	0	0	0	0	0
ITER Module 4	0	0	0	0	0
ITER Module 13	2	0	0	0	0
FNG Fusion Benchmark	63	0	0	0	0
ARIES First Wall	1	0	0	0	0
High Average Power Laser	0	0	0	0	0
Z-Pinch Fusion Reactor	3	0	0	0	0

Change in Number of Triangles

Table: The change ratio [*sealed*/*unsealed*] in the number of facets due to sealing is displayed as a function of facet tolerance [cm].

Model	Facet Tolerance [cm]				
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
UW Nuclear Reactor	0.71	0.99	1.00	1.00	1.01
Advanced Test Reactor	0.64	1.00	1.00	1.00	1.00
40° ITER Benchmark	1.00	1.01	1.02	1.02	1.03
ITER Test Blanket Module	0.90	1.00	1.01	1.01	1.01
ITER Module 4	0.65	0.98	1.00	1.01	1.01
ITER Module 13	0.78	1.00	1.00	1.00	1.00
FNG Fusion Benchmark	0.60	1.00	1.00	1.00	1.00
ARIES First Wall	1.00	1.00	1.00	1.00	1.00
High Average Power Laser	1.00	1.00	1.00	1.00	1.00
Z-Pinch Fusion Reactor	0.87	1.00	1.00	1.00	1.00

Inverted Facets

Table: The number of Faces containing inverted facets after sealing as a function of facet tolerance [cm].

Model	Facet Tolerance [cm]				
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
UW Nuclear Reactor	272	0	1	0	13
Advanced Test Reactor	30	0	0	0	0
40° ITER Benchmark	7	7	4	0	10
ITER Test Blanket Module	0	0	0	0	0
ITER Module 4	0	0	0	0	0
ITER Module 13	2	0	0	0	0
FNG Fusion Benchmark	16	0	0	0	0
ARIES First Wall	0	0	0	0	0
High Average Power Laser	0	0	0	0	0
Z-Pinch Fusion Reactor	2	1	0	0	0

Inverted facets occur when assumptions are not true:

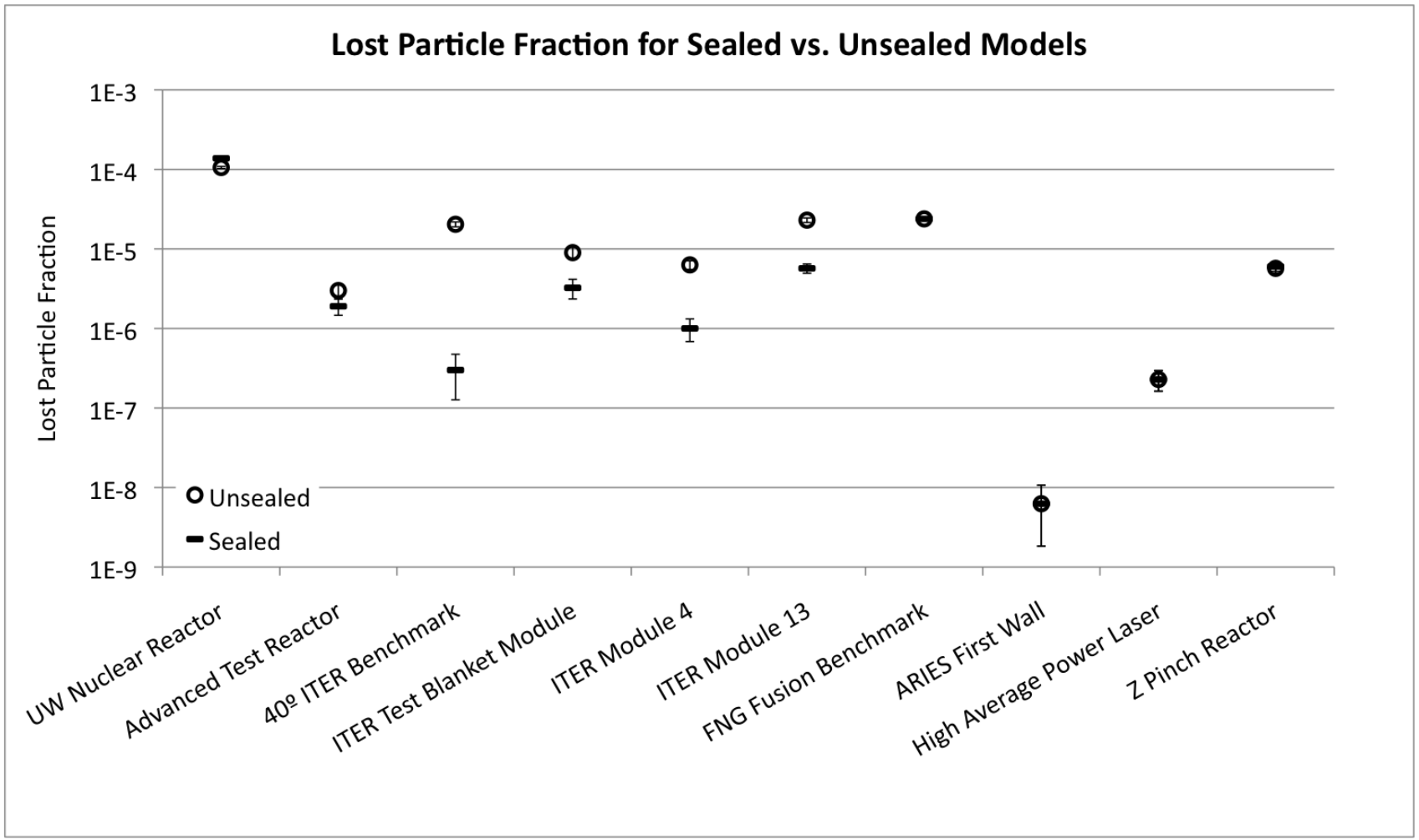
$\varepsilon_f > LFS$ or $\varepsilon_f < \varepsilon_m$.

Timing

Table: The time [seconds] to seal each model as a function of facet tolerance [cm], on one core of an Intel Xeon X5365 3.00 GHz CPU.

Model	Facet Tolerance [cm]				
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
UW Nuclear Reactor	136	65	64	156	587
Advanced Test Reactor	93	16	27	76	235
40° ITER Benchmark	6	12	38	71	236
ITER Test Blanket Module	15	9	9	14	30
ITER Module 4	10	8	8	23	67
ITER Module 13	6	5	6	19	67
FNG Fusion Benchmark	7	4	4	9	29
ARIES First Wall	1	3	5	13	36
High Average Power Laser	1	1	2	5	25
Z-Pinch Fusion Reactor	1	1	2	4	12

Application Example: Monte Carlo Radiation Transport



Leakage through unsealed surfaces is one cause of lost particles.

Conclusion

- Goals have been met: fast, automatic algorithm, supports non-manifold Faces, preserves input detail, minimizes new triangle creation
- Edges used to guide sealing, increasing robustness
- Successfully sealed 10 test models
- Implemented as open-source algorithm in MeshKit `http://trac.mcs.anl.gov/projects/fathom/wiki/MeshKit`

Acknowledgement

- Jason Kraftcheck, for advice and assistance using MOAB
- Sandia National Laboratories and the US ITER Project through Sandia Contracts 579323 and 866756
- US Department of Energy Scientific Discovery through Advanced Computing program under Contract DE-AC02-06CH11357

Questions?

bmsmith6@wisc.edu

References I

Amenta, N., Bern, M., and Kamvysselis, M. (1998).
A New Voronoi-Based Surface Reconstruction Algorithm.
In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 415–421, New York, NY, USA. ACM.

Barequet, G. and Kumar, S. (1997).
Repairing CAD Models.
In *VIS '97: Proceedings of the 8th Conference on Visualization '97*, Los Alamitos, CA, USA. IEEE Computer Society Press.

Barequet, G. and Sharir, M. (1995).
Filling Gaps in the Boundary of a Polyhedron.
Computer Aided Geometric Design, 12:207–229.

Borodin, P., Novotni, M., and Klein, R. (2002).
Progressive Gap Closing for Mesh Repairing.
Advances in Modelling, Animation and Rendering, pages 201–21.

Edelsbrunner, H. and Mücke, E. (1994).
Three-Dimensional Alpha Shapes.
ACM Transactions on Graphics, 13(1):43–72.

Ju, T. (2004).
Robust Repair of Polygonal Models.
In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 888–895, New York, NY, USA. ACM.

References II

Murali, T. M. and Funkhouser, T. A. (1997).

Consistent Solid and Boundary Representations from Arbitrary Polygonal Data.

In *I3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 155–ff., New York, NY, USA. ACM.

Newman, T. S. and Yi, H. (2006).

A Survey of the Marching Cubes Algorithm.

Computer and Graphics, 30:854–879.

Face Sealing Algorithm

I. $\forall F^i$
   a. $\forall \mathcal{V}^i \in_a \mathcal{F}^i$
      1. *find* $e \in \Omega F^i s.t.\ d(e, \mathcal{V}^i)$ *min*
      2. *if* $d(p \in_{adj} e, \mathcal{V}^i)$ *min, choose* p
      3. $p \rightarrow V^i$
   b. *group* $\Omega F^i \rightarrow \Omega^j F^i$ *using* $p \in \mathcal{V}^k \in_{adj} \mathcal{F}^i$
   c. $\forall \Omega^j F^i = e$
      1. *find* E^k *s.t.* $d_{oriented}(\Omega^j, E^k)$ *min*
      2. *if* E^k *not sealed yet,* $E^k \rightarrow \Omega^j F^i$
      3. *else seal*$(\Omega^j F^i, E^k)$

Arc Sealing Algorithm

Edge/arc sealing algorithm. $otherpoint(e, p)$ is the point adjacent to e not equal to p; for e_n, $p_n = next(\{e\}, e, p)$, e_n is the next edge along the ordered sequence of edges $\{e\}$ that shares p; $p_n = otherpoint(e_n, p)$.

Begin: seal arc $\Omega^j = \{p\}, \{e\}$, $E^k = \{p\}, \{e\}$

I. Initialize:
- a. $p_c = \Omega_{start}(\Omega^j)$
- b. $e_s,\ p_s = next(\Omega^j, -, p_c)$
- c. $e_e,\ p_e = next(E^k, -, p_c)$

II. while Ω^j not sealed
- a. if $p_e = p_s$
  1. $e_e \leftarrow e_s$
  2. Ω^j *sealed*.
- b. else if $d(p_s, p_e) \leq \epsilon_f$
  1. $p_e \leftarrow p_s$
  2. $e_e \leftarrow e_s$
  3. $p_c = p_e$
  4. $e_s,\ p_s = next(\Omega^j, e_e, p_c)$
  5. $e_e,\ p_e = next(E^k, e_e, p_c)$
- c. else if $d(p_e, e_s) \leq \epsilon_f$
  1. split e_s with p_e, $e_s \longrightarrow e_1(p_c, p_e), e_2(p_e, p_s)$
  2. $p_c = p_e$
  3. $e_e,\ p_e = next(E^k, e_e, p_c)$
  4. $e_s = e_2$

  (p_s unchanged)
- d. else if $d(p_s, e_e) \leq \epsilon_f$
  1. split e_e with p_s, $e_e \longrightarrow e_1(p_c, p_s), e_2(p_s, p_e)$
  2. $p_c = p_s$
  3. $e_s,\ p_s = next(\Omega^j, e_s, p_c)$
  4. $e_e = e_2$

  (p_e unchanged)

Two Approaches to Produce Watertight Faceting

Create new watertight faceting

- Discretize Edges
- Facet Faces constrained to Edges
- Pros: High success probability, access to solid model
- Cons: Complex implementation

Fix existing faceting flaws

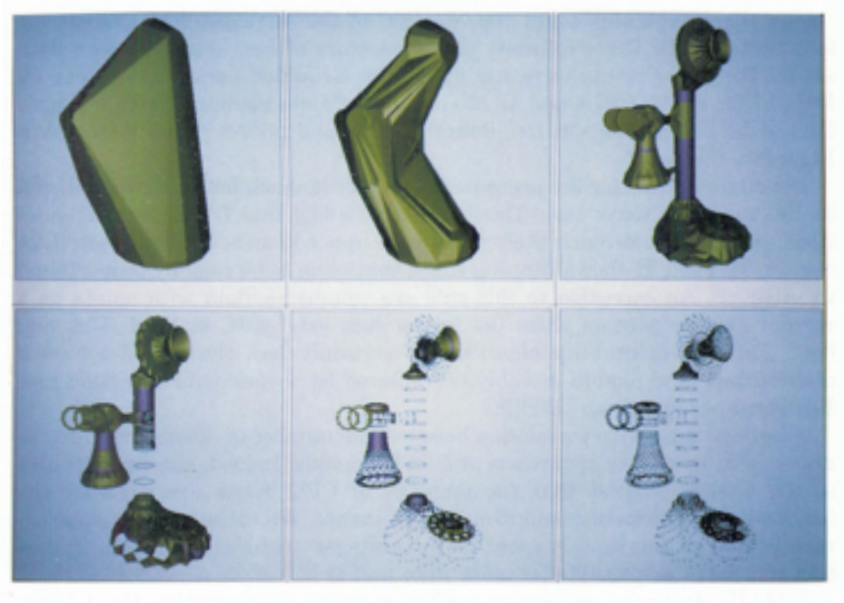
- Seal Faces to close gaps
- Pros: Fits with existing software chain, works for legacy models
- Cons: Must overcome loss of information from solid model

Problem Statement: Create algorithm to fix faceting flaws between Faces.

Algorithm Input Formats

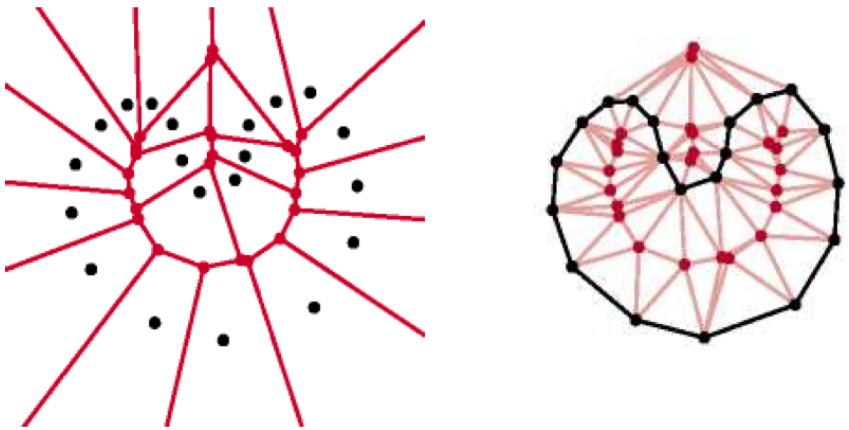
Point Data	Unorganized points on the Face of a solid
Polygon Data	Lists of polygons with (optional) normal vectors
Faceted CAD Data	Geometric topology with faceted Edges and Faces

Region-Based Approaches I



α-**Shapes** [Edelsbrunner and Mücke, 1994]

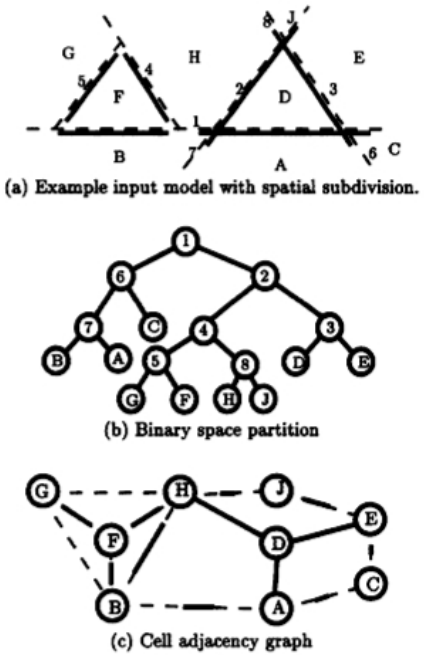
1. Input point data S
2. Compute Delaunay Triangularization (DT) of S
3. Compute circumradius r of each triangle
4. Compare with specified α
5. Triangles with $r < \alpha$ define boundary



Crust [Amenta et al., 1998]

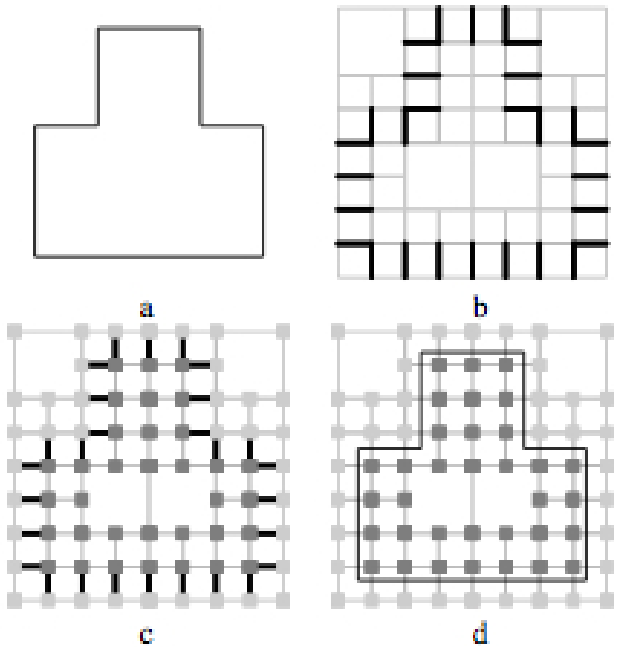
1. Input point data S
2. Compute DT of S
3. Compute Voronoi Diagram (VD) of DT
4. Let V be vertices of VD
5. Compute another DT of $S \cup V$
6. Edges with both points in S define boundary
7. VD medial axis filters unwanted edges of original DT

Region-Based Approaches II



Binary Space Partition [Murali and Funkhouser, 1997]

1. Input polygon data
2. Create BSP using polygon planes
3. Flag portion of plane coincident with input polygon as *opaque*
4. Calculate *solidity* of each cell in BSP
5. Boundary is defined by cells of positive solidity



Marching Cubes [Ju, 2004]

1. Input polygon data
2. Create regular lattice of edges e, points p, and cubes c
3. Find edges e that intersect input polygons
4. Mark intersection edge endpoints p as *in* or *out*
5. Infer signs on unmarked points p
6. Each cube is defined by sign of its points p
7. Determine boundary of union of cubes

Goal: Recreate Missing Information

Table: Overview of defect healing methods for various algorithms.

Algorithm	**Heals defects by...**
α-Shapes	▶ Select proper α to resolve features without penetrating Delaunay triangulation of surface
Crust	▶ Use heuristic to label poles inside/outside based on local inside/outside characteristics
BSP	▶ Extends input 2D polygonal boundary to planar boundary of convex 3D cells. Matrix equation used to label cells as inside/outside based on shared cell boundaries
Marching Cubes	▶ Select single isosurface at a distance far enough to bridge defects. ▶ Select range of isosurfaces to identify entire cubes as inside/outside [Newman and Yi, 2006] ▶ Triangulate patch across gaps to minimizes area or other heuristic
All Mesh-Based	▶ Triangulate patch across gaps to minimizes area or other heuristic ▶ Vertex-vertex contraction by proximity ▶ Vertex-edge contraction by proximity