



3D “Robust” Delaunay Refinement

Steven E. Pav (with Noel J. Walkington)

University of California, San Diego



Quality Delaunay Refinement

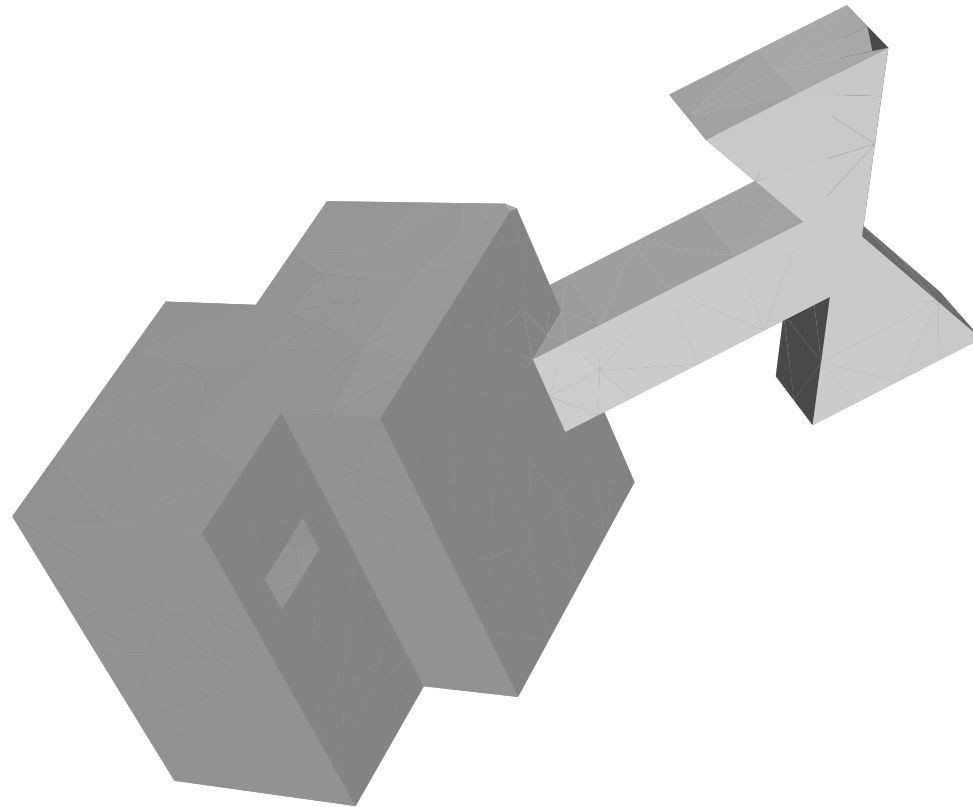
- Meshing a Piecewise Linear Complex (PLC) in \mathbb{R}^3 .

Quality Delaunay Refinement

- Meshing a Piecewise Linear Complex (PLC) in \mathbb{R}^3 .
 - Conforming Delaunay mesh.
 - Quality tetrahedra.
(bounded circumradius / shortest-edge ratio)

Quality Delaunay Refinement

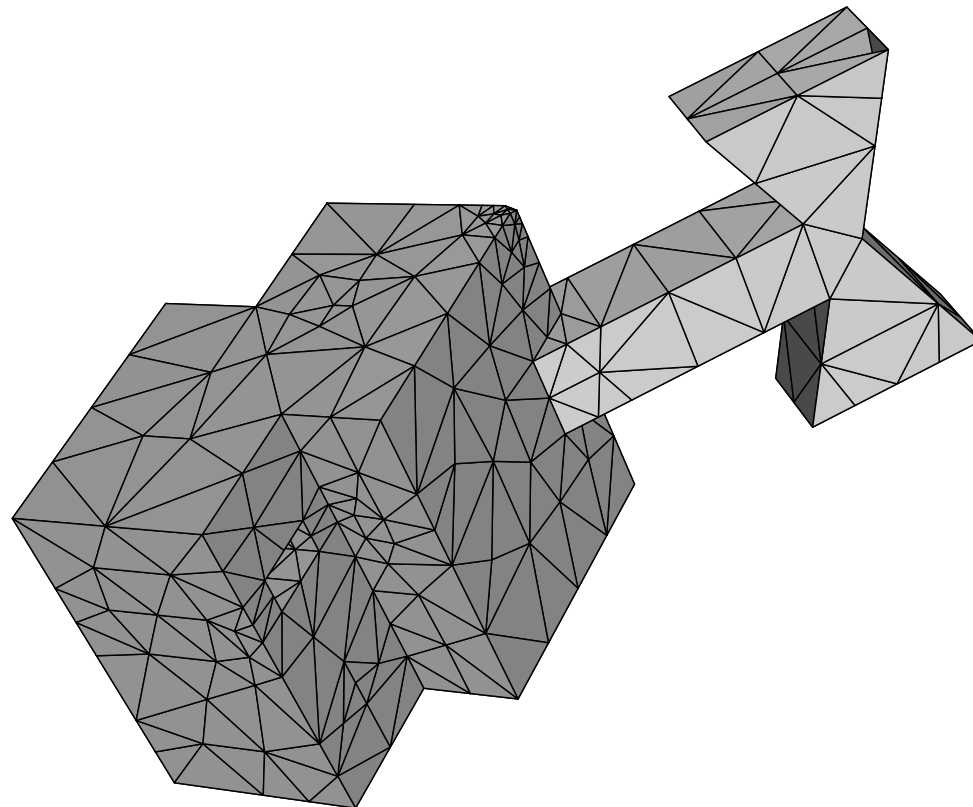
- Meshing a Piecewise Linear Complex (PLC) in \mathbb{R}^3 .



3D Widget Input

Quality Delaunay Refinement

- Meshing a Piecewise Linear Complex (PLC) in \mathbb{R}^3 .



3D Widget Output

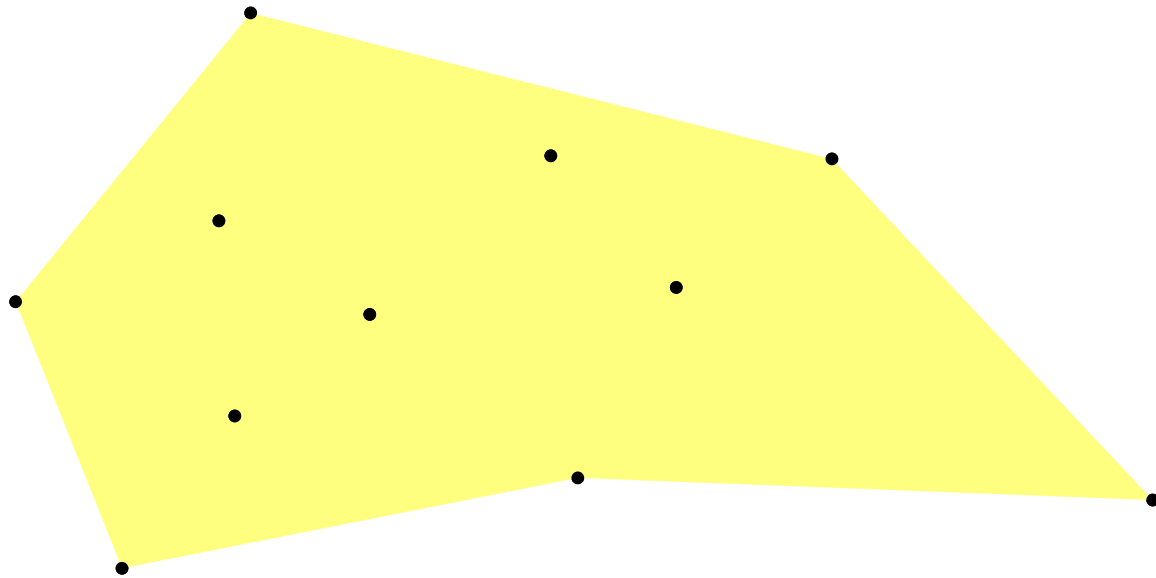
3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.

3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.

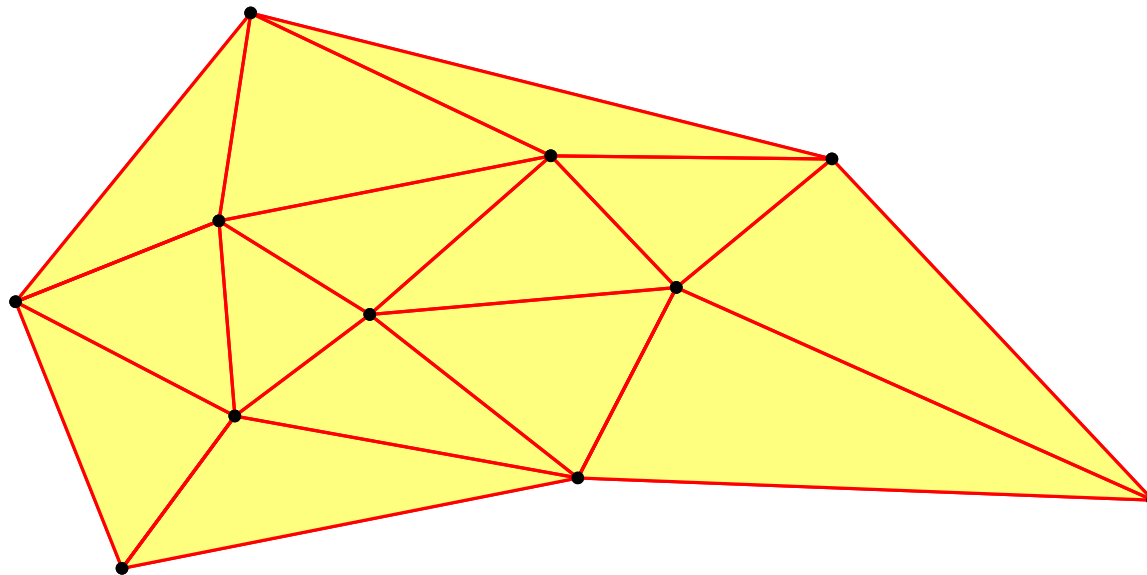
$$\mathcal{F} = \bigcup_{f \text{ a face of PLC}} \mathcal{D}(\mathcal{P} \cap f)$$



3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.

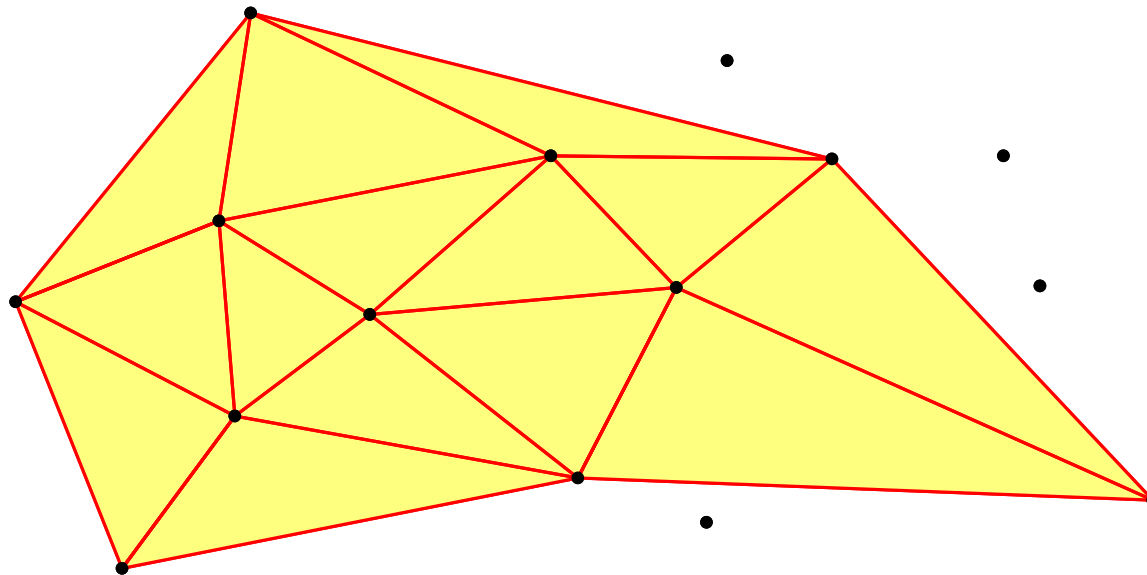
$$\mathcal{F} = \bigcup_{f \text{ a face of PLC}} \mathcal{D}(\mathcal{P} \cap f)$$



3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.

$$\mathcal{F} = \bigcup_{f \text{ a face of PLC}} \mathcal{D}(\mathcal{P} \cap f)$$



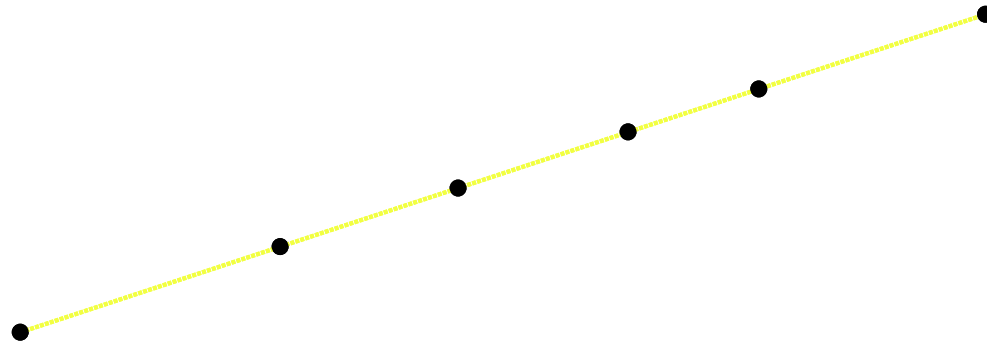
3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).

3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).

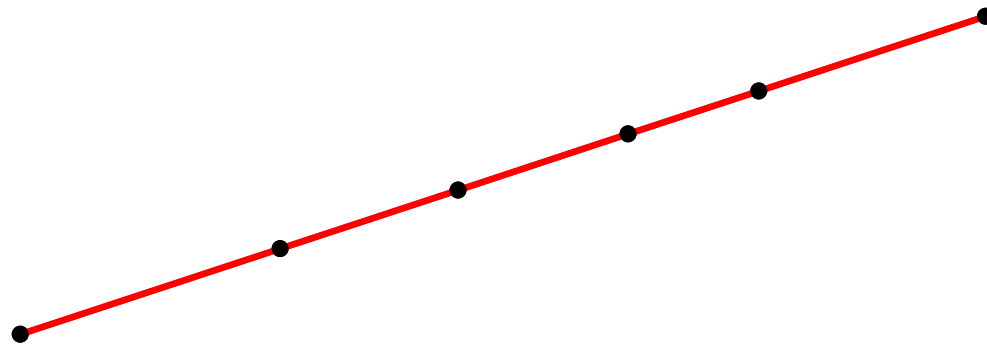
$$\mathcal{S} = \bigcup_{s \text{ a seg of PLC}} \mathcal{D}(\mathcal{P} \cap s)$$



3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).

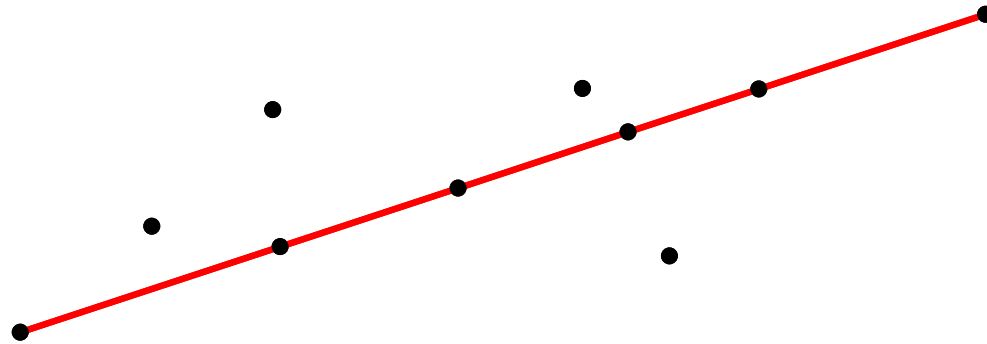
$$\mathcal{S} = \bigcup_{s \text{ a seg of PLC}} \mathcal{D}(\mathcal{P} \cap s)$$



3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).

$$\mathcal{S} = \bigcup_{s \text{ a seg of PLC}} \mathcal{D}(\mathcal{P} \cap s)$$



3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).
- \mathcal{S} defined by \mathcal{P} (and PLC).

3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).
- \mathcal{S} defined by \mathcal{P} (and PLC).
- Initialize \mathcal{P} as input points.
Add points to \mathcal{P} to:
 1. Make segments of \mathcal{S} appear in $\mathcal{D}(\mathcal{P})$.

3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).
- \mathcal{S} defined by \mathcal{P} (and PLC).
- Initialize \mathcal{P} as input points.
Add points to \mathcal{P} to:
 1. Make segments of \mathcal{S} appear in $\mathcal{D}(\mathcal{P})$.
 2. Make facets of \mathcal{F} appear in $\mathcal{D}(\mathcal{P})$.

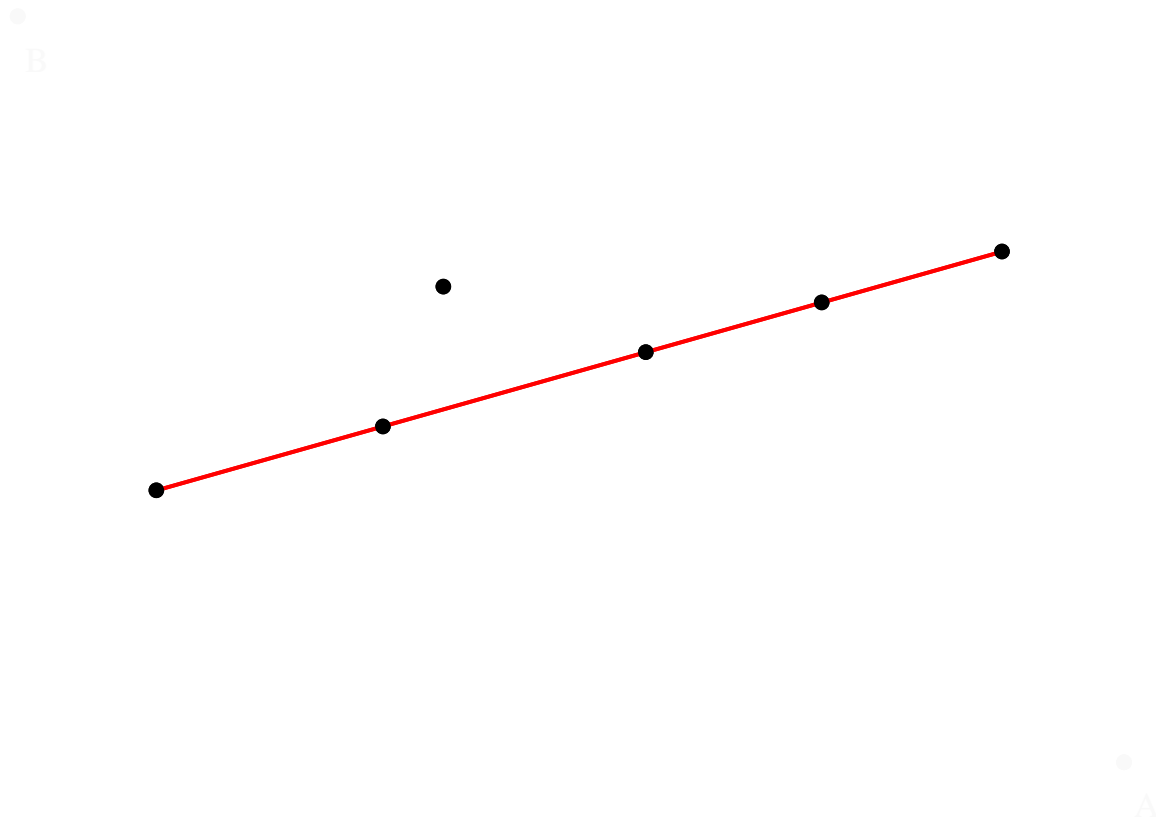
3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).
- \mathcal{S} defined by \mathcal{P} (and PLC).
- Initialize \mathcal{P} as input points.
Add points to \mathcal{P} to:
 1. Make segments of \mathcal{S} appear in $\mathcal{D}(\mathcal{P})$.
 2. Make facets of \mathcal{F} appear in $\mathcal{D}(\mathcal{P})$.
 3. Ensure quality of tets in $\mathcal{D}(\mathcal{P})$.

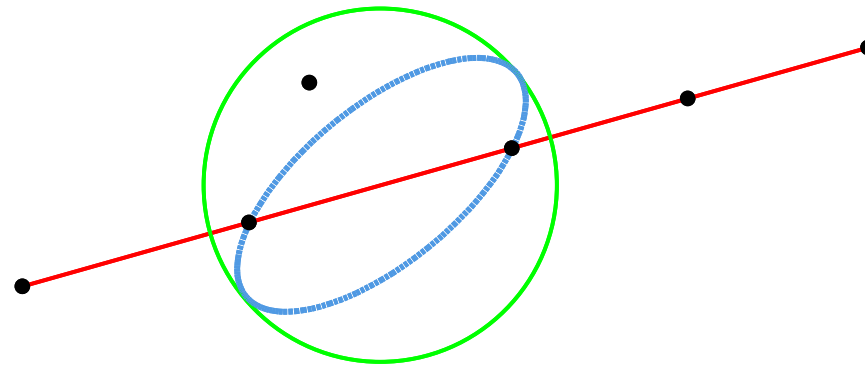
3D Delaunay Refinement

- Maintain set of points, segments, facets, $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.
- \mathcal{F} defined by \mathcal{P} (and PLC).
- \mathcal{S} defined by \mathcal{P} (and PLC).
- Initialize \mathcal{P} as input points.
Add points to \mathcal{P} to:
 1. Make segments of \mathcal{S} appear in $\mathcal{D}(\mathcal{P})$.
 2. Make facets of \mathcal{F} appear in $\mathcal{D}(\mathcal{P})$.
 3. Ensure quality of tets in $\mathcal{D}(\mathcal{P})$.
- Return Delaunay Tetrahedralization of \mathcal{P} .

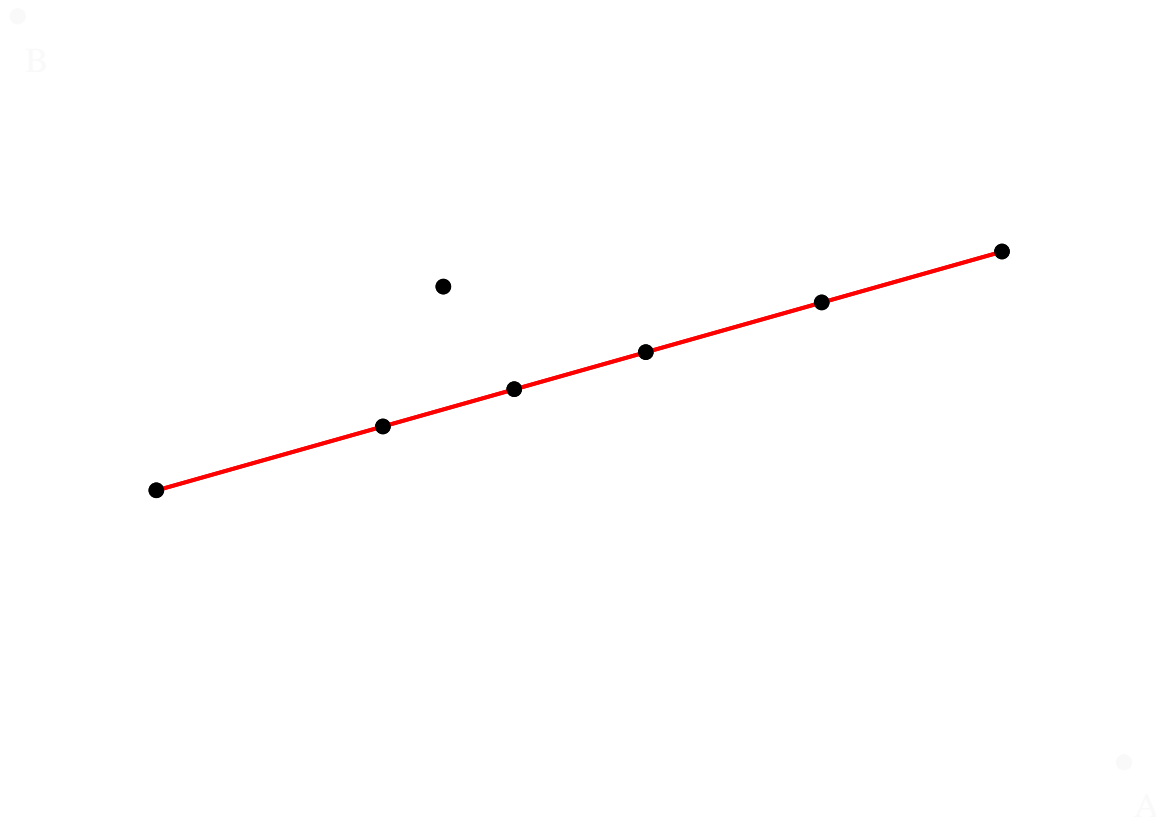
1. **Split** encroached segment.



1. **Split** encroached segment.

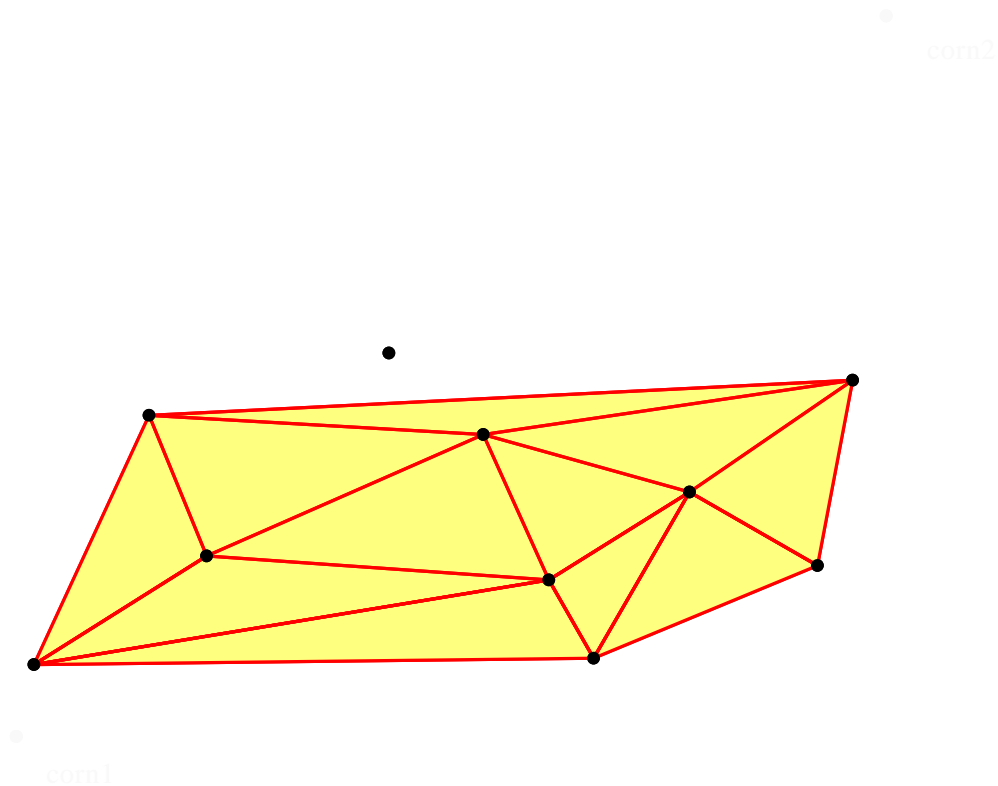


1. **Split** encroached segment.



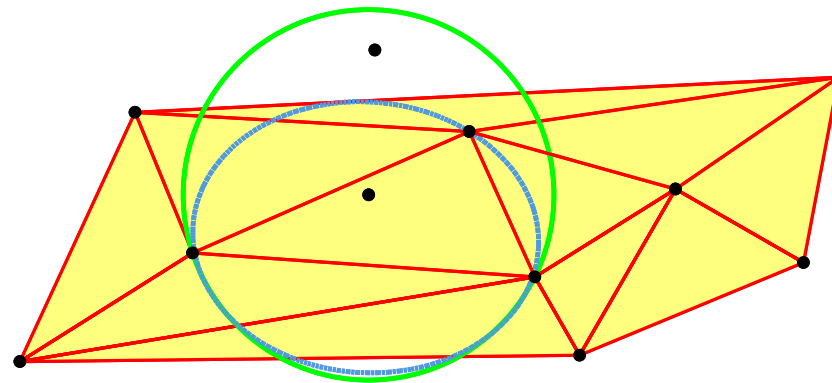
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.



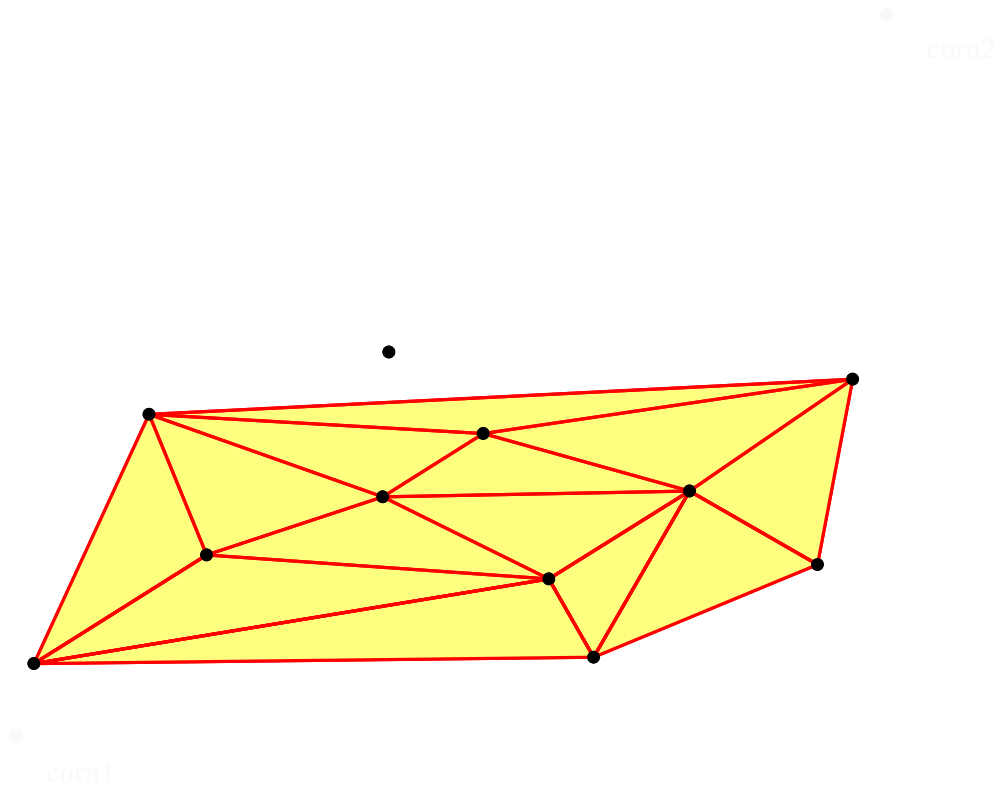
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.



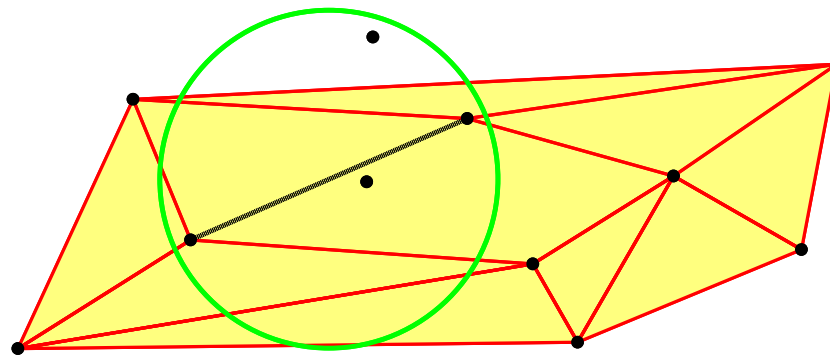
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.



3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
Unless circumcenter encroaches a segment.
(Split that segment instead)

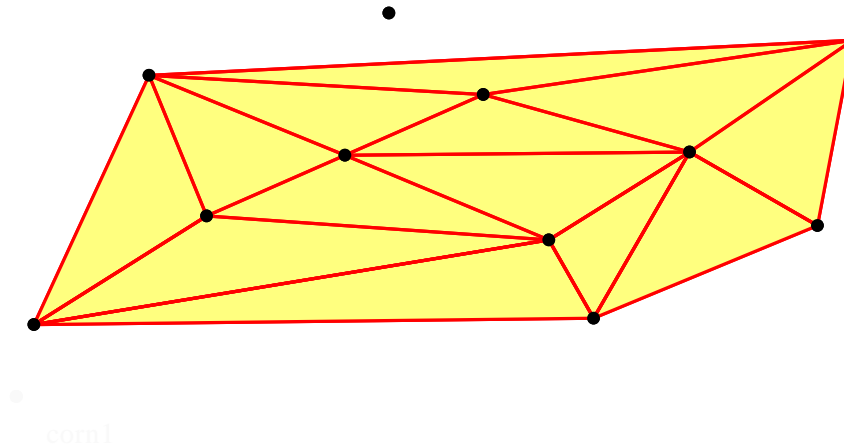


com1

com2

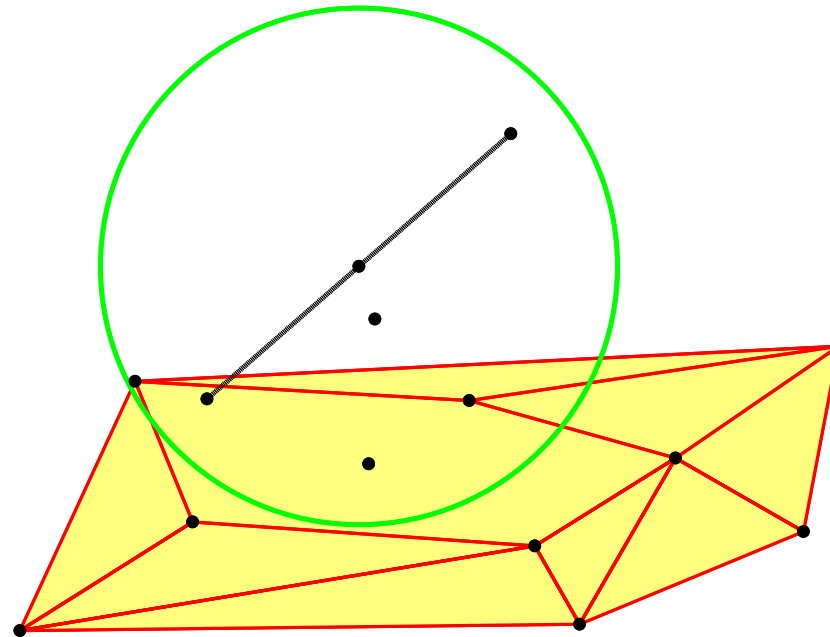
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
Unless circumcenter encroaches a segment.
(Split that segment instead)



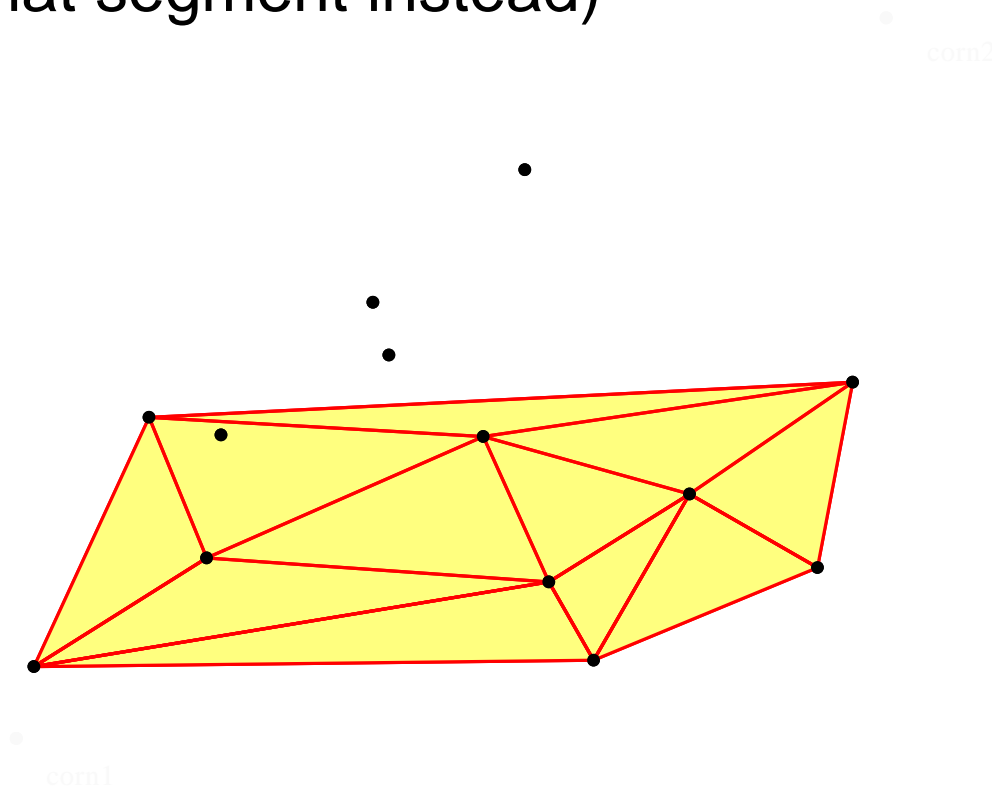
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
Unless circumcenter encroaches a segment.
 (Split that segment instead)



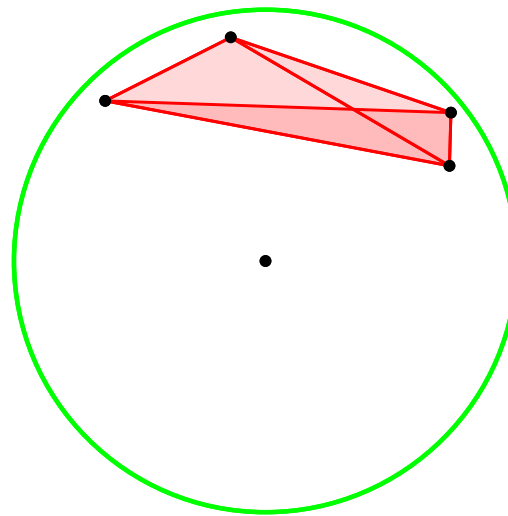
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
Unless circumcenter encroaches a segment.
(Split that segment instead)



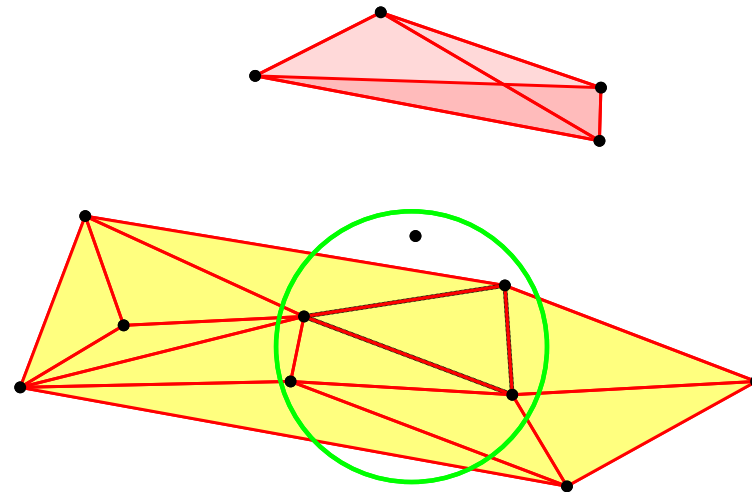
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
3. Add circumcenter of poor-quality Delaunay tet.
(Radius / shortest-edge $> B$)



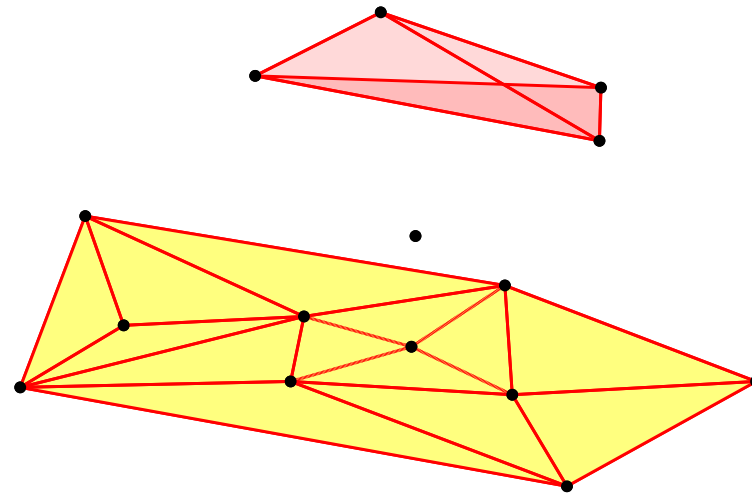
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
3. Add circumcenter of poor-quality Delaunay tet.
 (Radius / shortest-edge $> B$)
Unless it encroaches a facet or segment.
 Attempt the **split** instead.



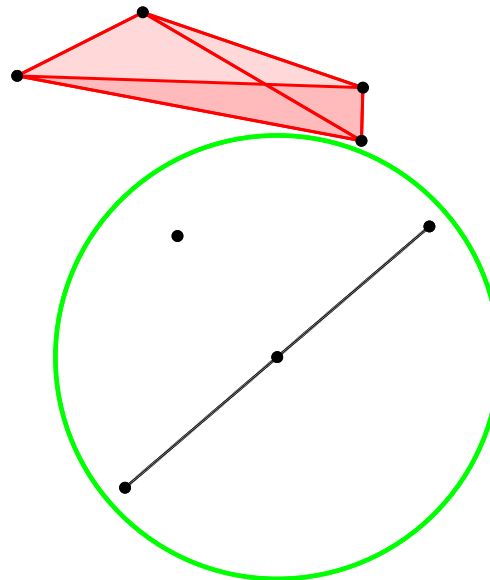
3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
3. Add circumcenter of poor-quality Delaunay tet.
 (Radius / shortest-edge $> B$)
Unless it encroaches a facet or segment.
 Attempt the **split** instead.



3D Delaunay Refinement (II)

1. **Split** encroached segment.
2. **Split** encroached facet.
3. Add circumcenter of poor-quality Delaunay tet.
 (Radius / shortest-edge $> B$)
Unless it encroaches a facet or segment.
 Attempt the **split** instead.



Delaunay Refinement Algorithm

If algorithm terminates:

- No encroached segments in \mathcal{S} .
- No encroached facets in \mathcal{F} .
- No Delaunay tet with circumradius/edge $> B$.

Delaunay Refinement Algorithm

If algorithm terminates:

- No encroached segments in \mathcal{S} .
- No encroached facets in \mathcal{F} .
- No Delaunay tet with circumradius/edge $> B$.

Classical Termination Proof:

- Define $\text{lfs}(x)$ as “radius of smallest closed ball with center x intersecting 2 disjoint features of the input.”

Delaunay Refinement Algorithm

If algorithm terminates:

- No encroached segments in \mathcal{S} .
- No encroached facets in \mathcal{F} .
- No Delaunay tet with circumradius/edge $> B$.

Classical Termination Proof:

- Define $\text{lfs}(x)$ as “radius of smallest closed ball with center x intersecting 2 disjoint features of the input.”
- When q to be added to \mathcal{P} , show

$$\text{lfs}(q) \leq G_i |q - \mathcal{P}|$$

(“Well Graded” mesh)

Delaunay Refinement Algorithm

If algorithm terminates:

- No encroached segments in \mathcal{S} .
- No encroached facets in \mathcal{F} .
- No Delaunay tet with circumradius/edge $> B$.

Classical Termination Proof:

- Define $\text{lfs}(x)$ as “radius of smallest closed ball with center x intersecting 2 disjoint features of the input.”
- When q to be added to \mathcal{P} , show

$$0 < \text{lfs}_{\min} \leq \text{lfs}(q) \leq G_i |q - \mathcal{P}|$$

(“Well Graded” mesh)

Delaunay Refinement Algorithm

If algorithm terminates:

- No encroached segments in \mathcal{S} .
- No encroached facets in \mathcal{F} .
- No Delaunay tet with circumradius/edge $> B$.

Classical Termination Proof:

- Define $\text{ifs}(x)$ as “radius of smallest closed ball with center x intersecting 2 disjoint features of the input.”
- When q to be added to \mathcal{P} , show

$$0 < \text{ifs}_{\min} \leq G_i |q - \mathcal{P}|$$

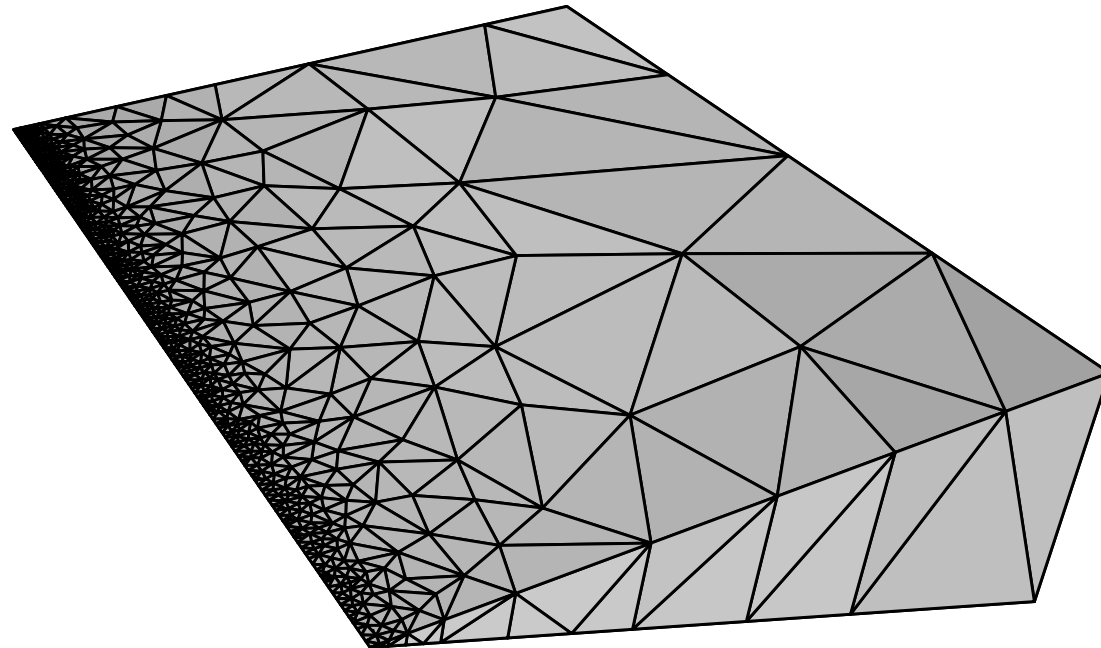
(“Weakly Graded” mesh)



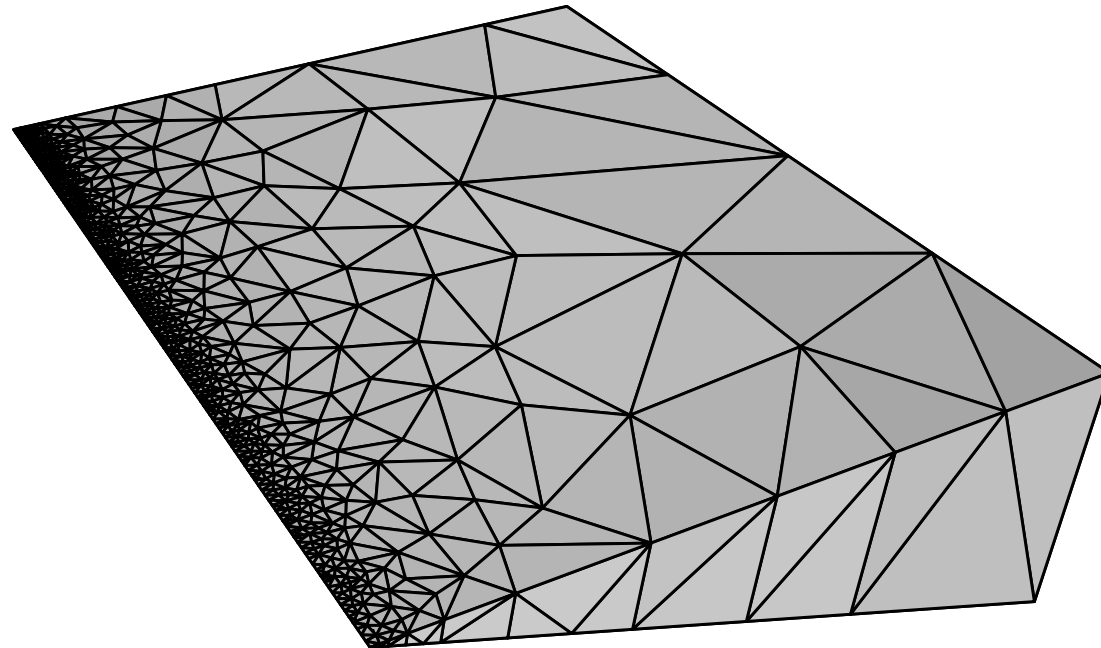
An Incomplete History of 3D Delaunay Refinement

- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.

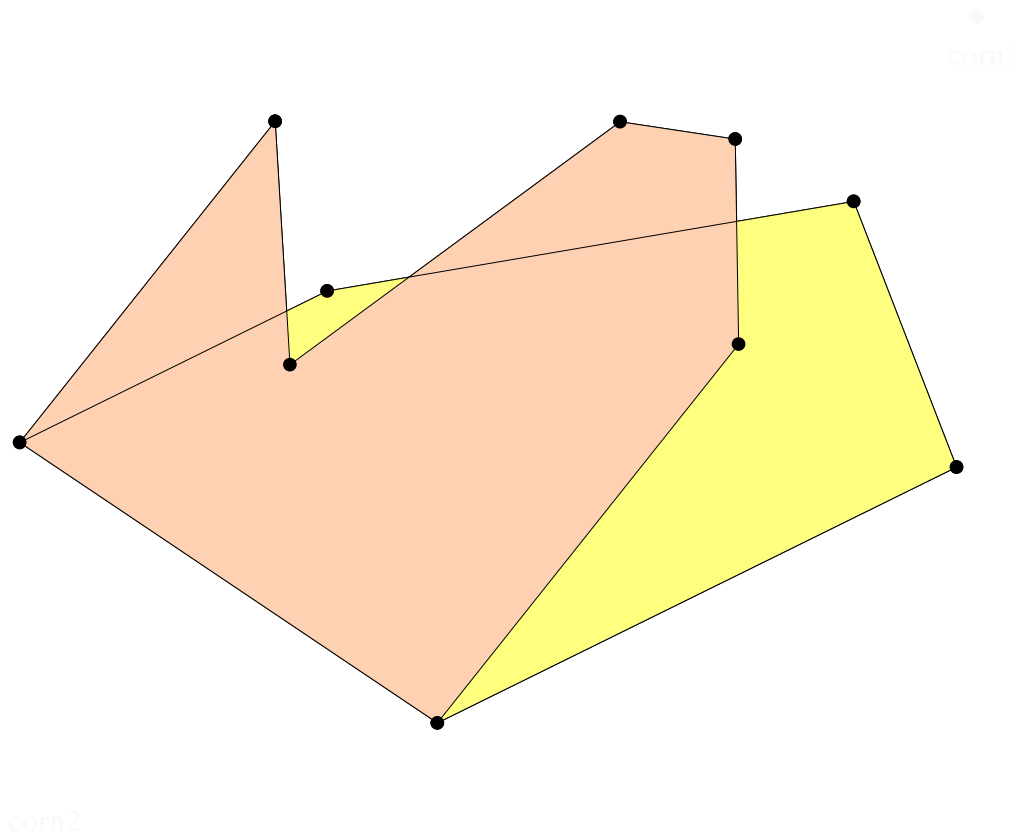
- Infinite cascade of point additions.



- Infinite cascade of point additions.
- Caused by small angles in the input.

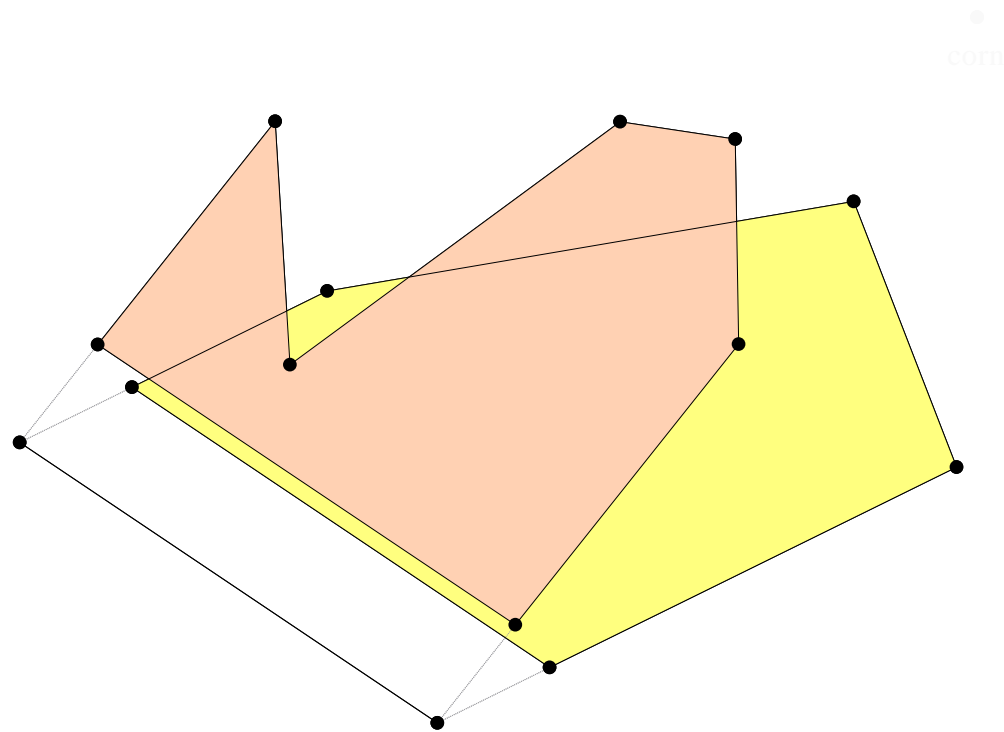


- Divide into “free” area, and “collar” or “buffer.”



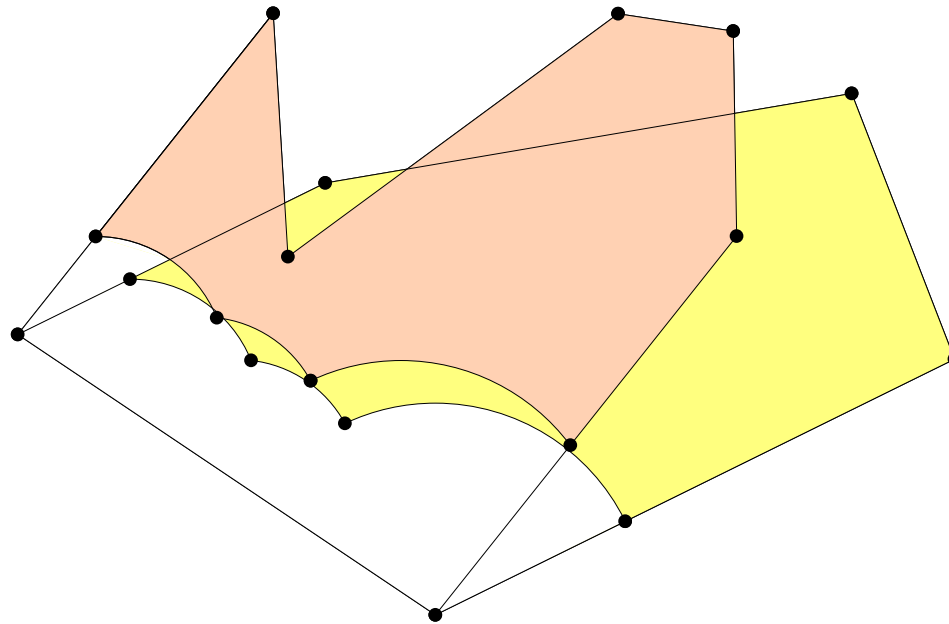
3D Corner Lopping

- Divide into “free” area, and “collar” or “buffer.”
- Make adjacent faces disjoint.



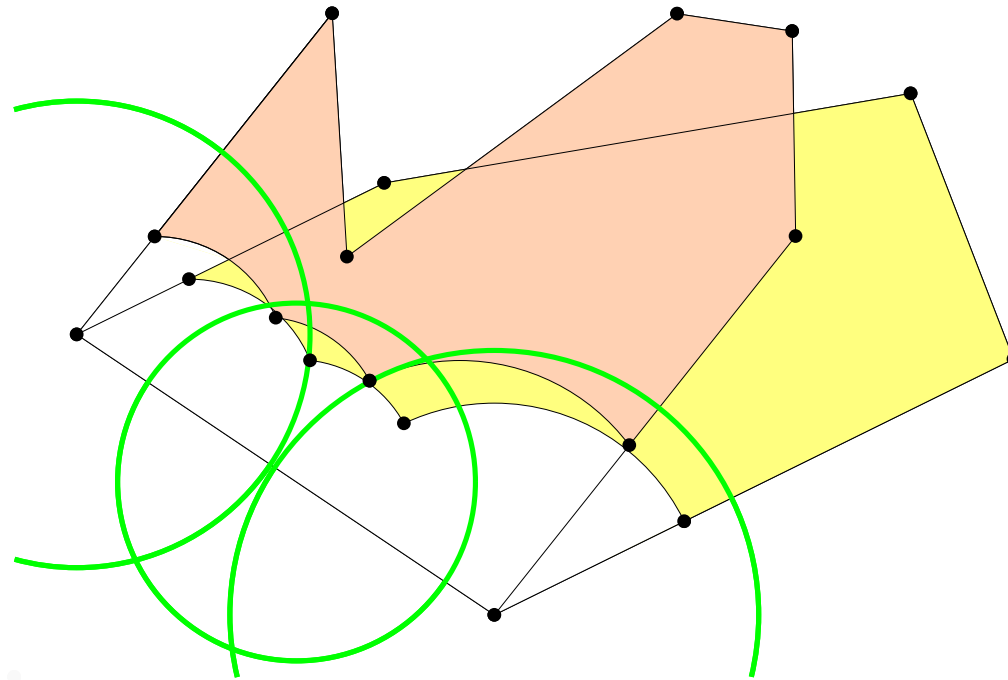
3D Corner Lopping

- Divide into “free” area, and “collar” or “buffer.”
- Make adjacent faces disjoint.
- Cut faces “adaptively.”



3D Corner Lopping

- Divide into “free” area, and “collar” or “buffer.”
- Make adjacent faces disjoint.
- Cut faces “adaptively.”
- Determine face cuts by a ball covering of edges.





An Incomplete History of 3D Delaunay Refinement

- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.
- Murphy *et al.* (2000): corner lopping in 3D
 $\theta^* > 0$, $B = \infty$. Uniform meshes.

An Incomplete History of 3D Delaunay Refinement

- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.
- Murphy *et al.* (2000): corner lopping in 3D
 $\theta^* > 0$, $B = \infty$. Uniform meshes.
- Cohen-Steiner *et al.* (2002): $\theta^* > 0$, $B = \infty$.
Precompute lfs; uses circular arcs.

An Incomplete History of 3D Delaunay Refinement

- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.
- Murphy *et al.* (2000): corner lopping in 3D
 $\theta^* > 0$, $B = \infty$. Uniform meshes.
- Cohen-Steiner *et al.* (2002): $\theta^* > 0$, $B = \infty$.
Precompute lfs; uses circular arcs.
- Cheng & Poon (2003): $\theta^* > 0$, $B > 16$.
Precompute lfs; uses spherical patches.

An Incomplete History of 3D Delaunay Refinement

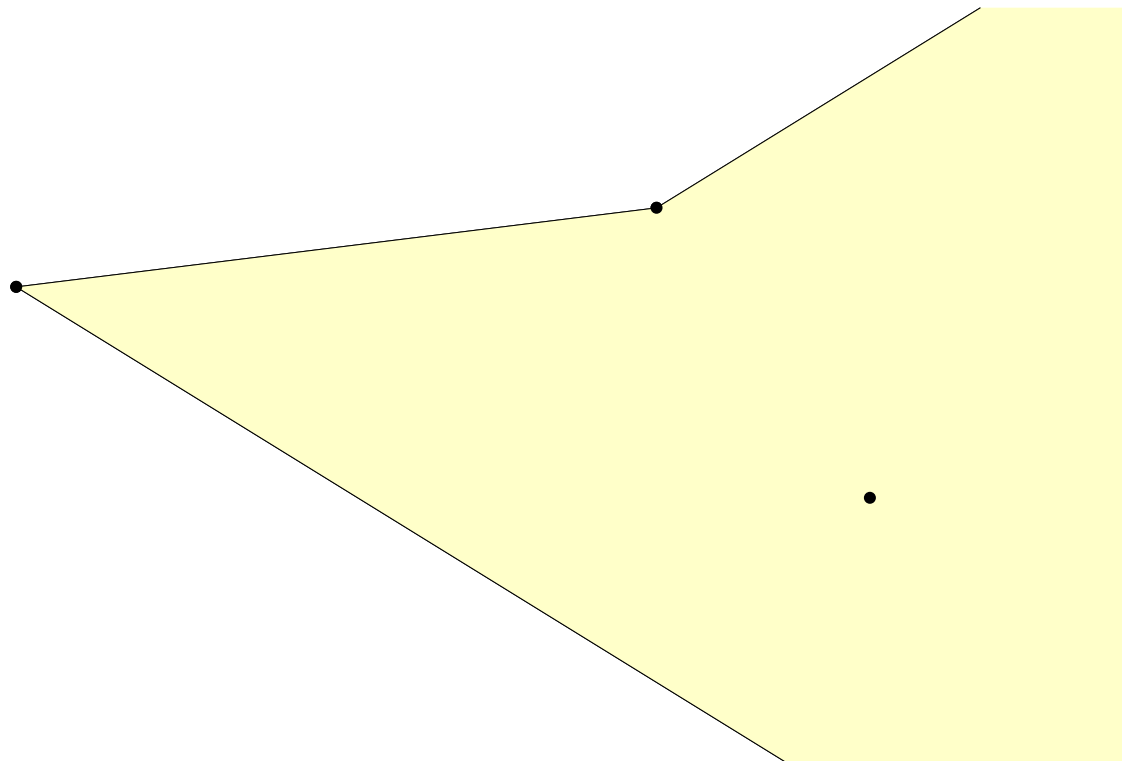
- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.
- Murphy *et al.* (2000): corner lopping in 3D
 $\theta^* > 0$, $B = \infty$. Uniform meshes.
- Cohen-Steiner *et al.* (2002): $\theta^* > 0$, $B = \infty$.
Precompute lfs; uses circular arcs.
- Cheng & Poon (2003): $\theta^* > 0$, $B > 16$.
Precompute lfs; uses spherical patches.
- Cheng *et al.* (2004): $\theta^* > 0$, $B \gtrsim 3.41$.
Precompute lfs; Polyhedral input only.

An Incomplete History of 3D Delaunay Refinement

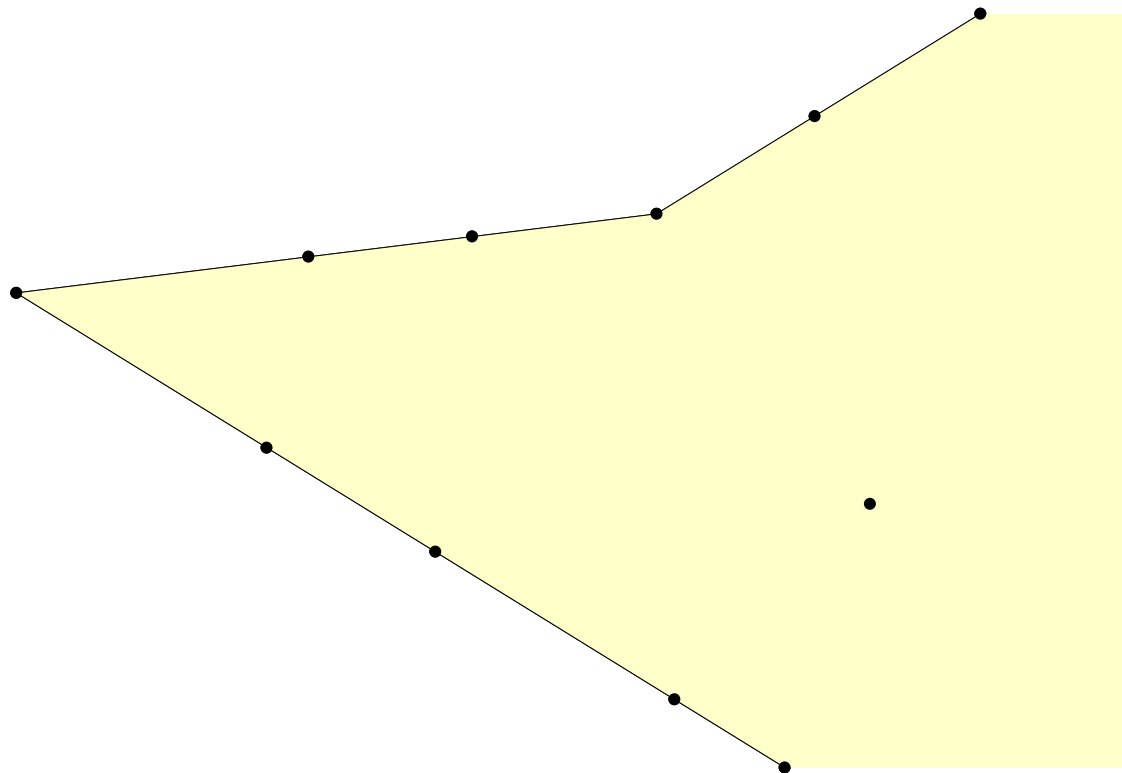
- Shewchuk (1997): Ruppert's Algorithm in 3D.
Input angle $\theta^* > \pi/2$, output radius-edge ratio $B > 2$.
- Murphy *et al.* (2000): corner lopping in 3D
 $\theta^* > 0$, $B = \infty$. Uniform meshes.
- Cohen-Steiner *et al.* (2002): $\theta^* > 0$, $B = \infty$.
Precompute lfs; uses circular arcs.
- Cheng & Poon (2003): $\theta^* > 0$, $B > 16$.
Precompute lfs; uses spherical patches.
- Cheng *et al.* (2004): $\theta^* > 0$, $B \gtrsim 3.41$.
Precompute lfs; Polyhedral input only.

In 2D, lfs computation not necessary.

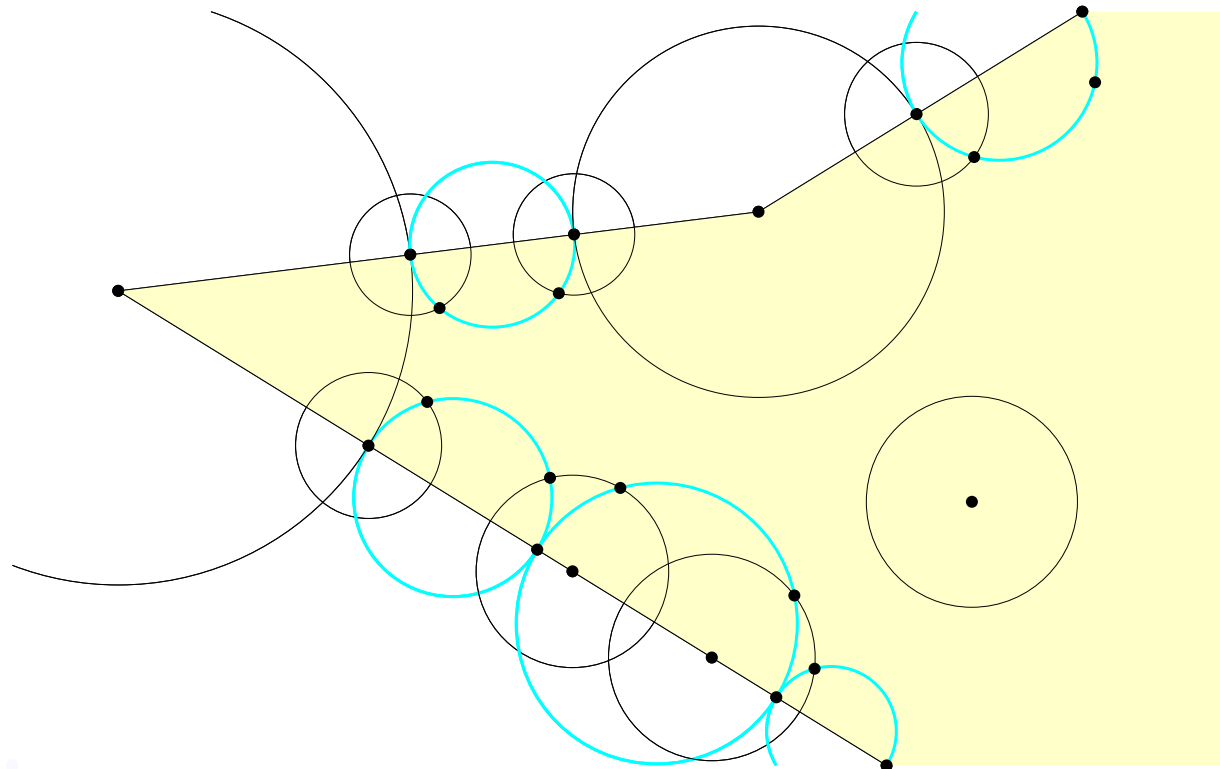
- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.



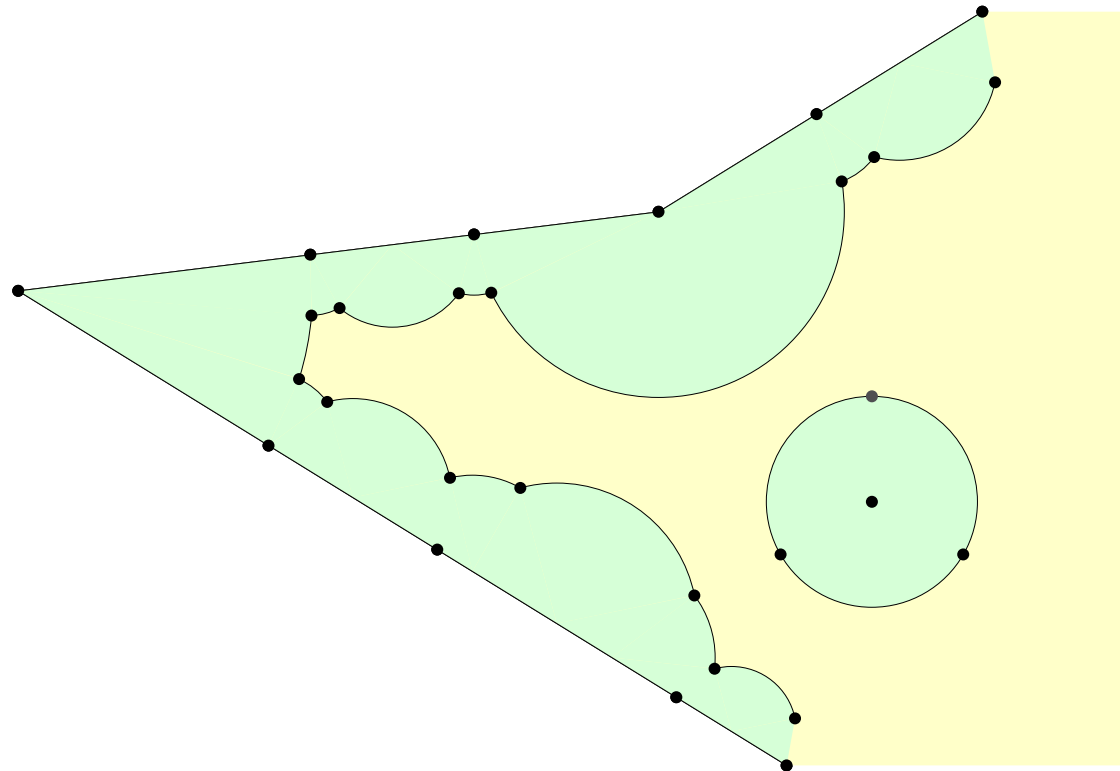
- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.



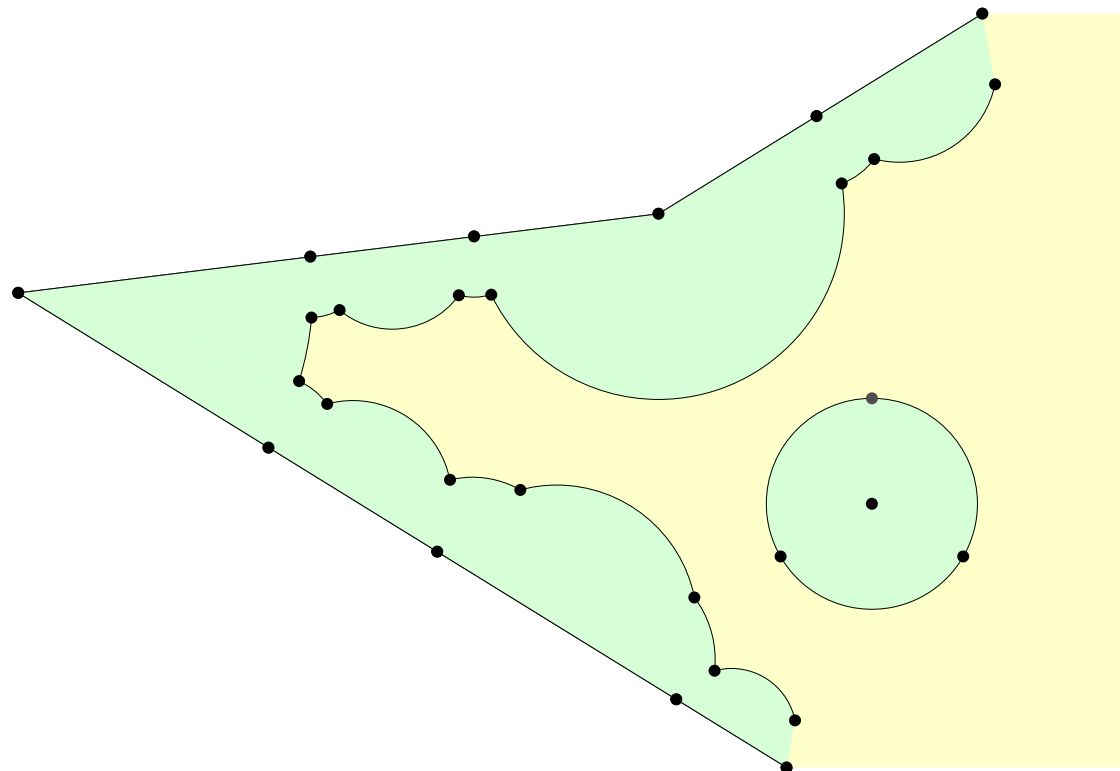
- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.



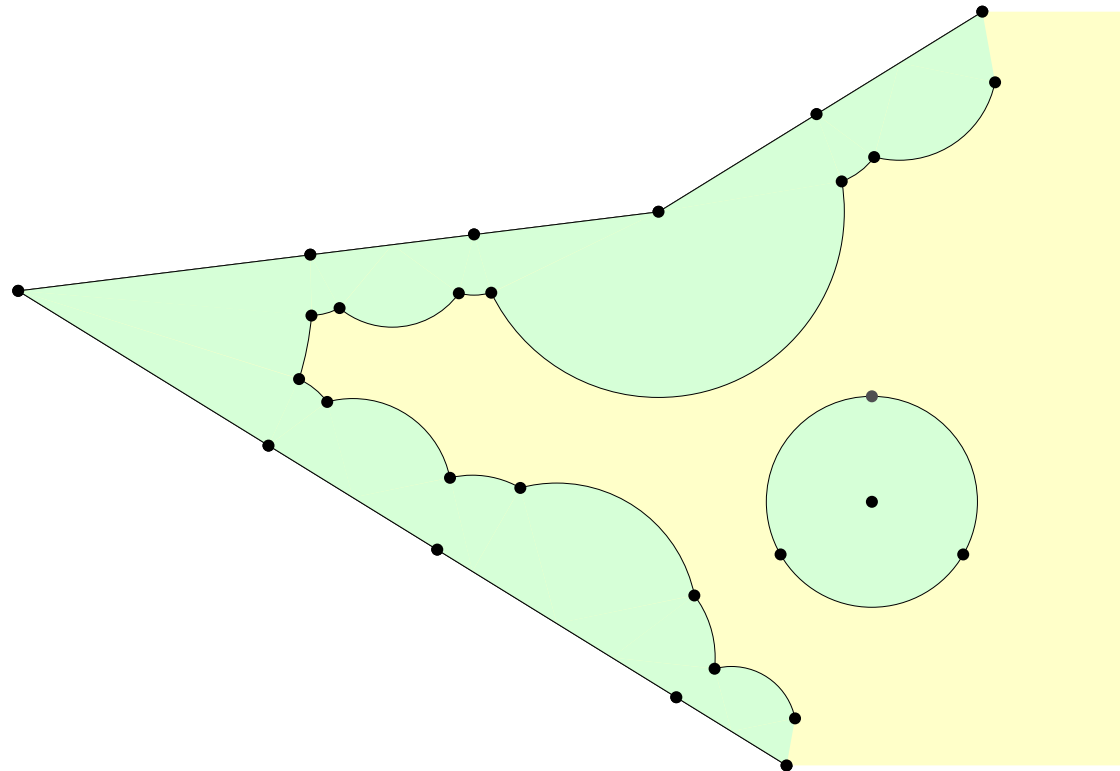
- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.



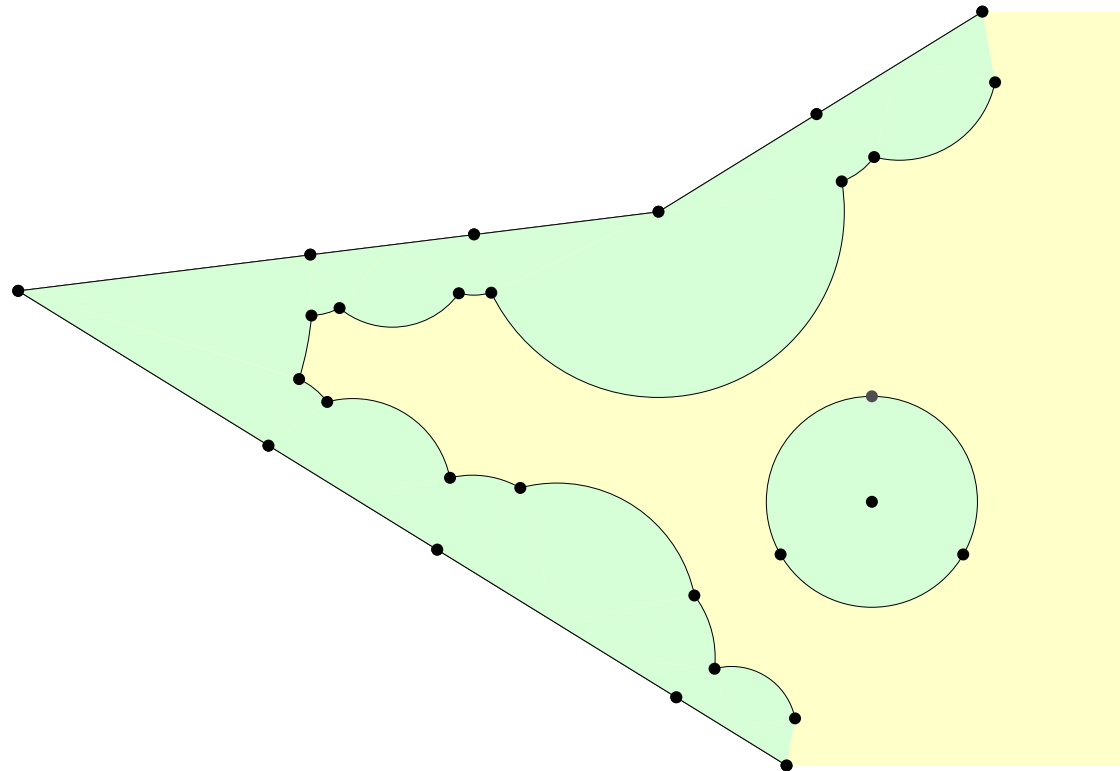
- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.
- Arcs meet at **obtuse** angles.



- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.
- Arcs meet at **obtuse** angles.
- “Disjoint” arcs do not overlap.



- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.
- Arcs meet at **obtuse** angles.
- “Disjoint” arcs do not overlap.
- Lengths of arcs bounded by l_{fs} .



- Maintain arcs: $(\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{F})$.
- Arcs meet at **obtuse** angles.
- “Disjoint” arcs do not overlap.
- Lengths of arcs bounded by l_{fs} .
- Arcs constructed by local searches.
(Computation of l_{fs} **not** required.)



The Algorithm

- Construct the collar region.

- Construct the collar region.
 1. If an arc is encroached, split it.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.
 - arc: split the arc.
 - segment: split segment, rebuild collar.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.
 - arc: split the arc.
 - segment: split segment, rebuild collar.
 4. Split bad Delaunay Tet. (Use any $B > 2$)

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.
 - arc: split the arc.
 - segment: split segment, rebuild collar.
 4. Split bad Delaunay Tet. (Use any $B > 2$)
Unless encroaches arc, facet, or collar.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.
 - arc: split the arc.
 - segment: split segment, rebuild collar.
 4. Split bad Delaunay Tet. (Use any $B > 2$)
Unless encroaches arc, facet, or collar.
 - arc, facet: split the arc or facet.
 - collar: leave the tet, make no split.

- Construct the collar region.
 1. If an arc is encroached, split it.
 2. If collar is encroached, split segments, rebuild collar.
 3. If a facet is encroached, split it.
Unless circumcenter encroaches an arc or segment.
 - arc: split the arc.
 - segment: split segment, rebuild collar.
 4. Split bad Delaunay Tet. (Use any $B > 2$)
Unless encroaches arc, facet, or collar.
 - arc, facet: split the arc or facet.
 - collar: leave the tet, make no split.
- Bad tets may remain near the collar.

- Implementing the algorithm.

Future Work

- Implementing the algorithm.
- Find optimal control parameters.

- Implementing the algorithm.
- Find optimal control parameters.
- Get quality bound on bad tets. (Analogous to 2D)

- Implementing the algorithm.
- Find optimal control parameters.
- Get quality bound on bad tets. (Analogous to 2D)
- Make collar “adaptive” to small dihedrals.

- Implementing the algorithm.
- Find optimal control parameters.
- Get quality bound on bad tets. (Analogous to 2D)
- Make collar “adaptive” to small dihedrals.
- Eliminate protecting arcs? (Make collar “implicit.”)

- [1] C. Boivin and C. F. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185–1213, 2002.
- [2] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Quality meshing for polyhedra with small angles. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry (Brooklyn, New York)*. ACM, June 2004.
- [3] S.-W. Cheng and S.-H. Poon. Graded conforming Delaunay tetrahedralization with bounded radius-edge ratio. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003)*, pages 295–304, New York, 2003. ACM.
- [4] D. Cohen-Steiner, E. Colin de Verdière, and M. Yvinec. Conforming Delaunay triangulations in 3d. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 2002.
- [5] S. E. Pav. *Delaunay Refinement Algorithms*. PhD thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 2003.
- [6] J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997. Available as Technical Report CMU-CS-97-137.