Computing Straight Skeletons by Means of Kinetic Triangulations

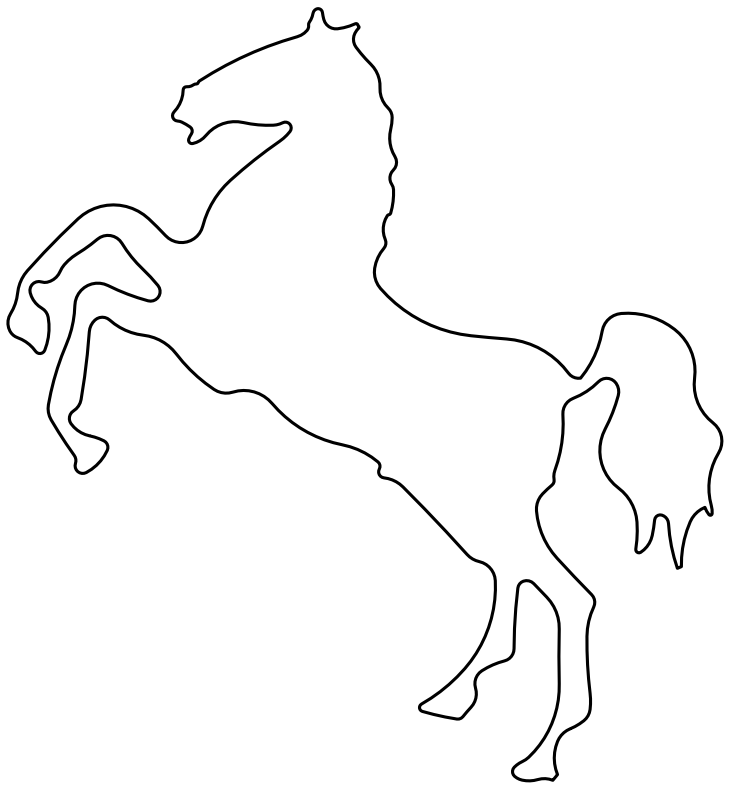
Peter Palfrader

October 2013

Computing Straight Skeletons by Means of Kinetic Triangulations

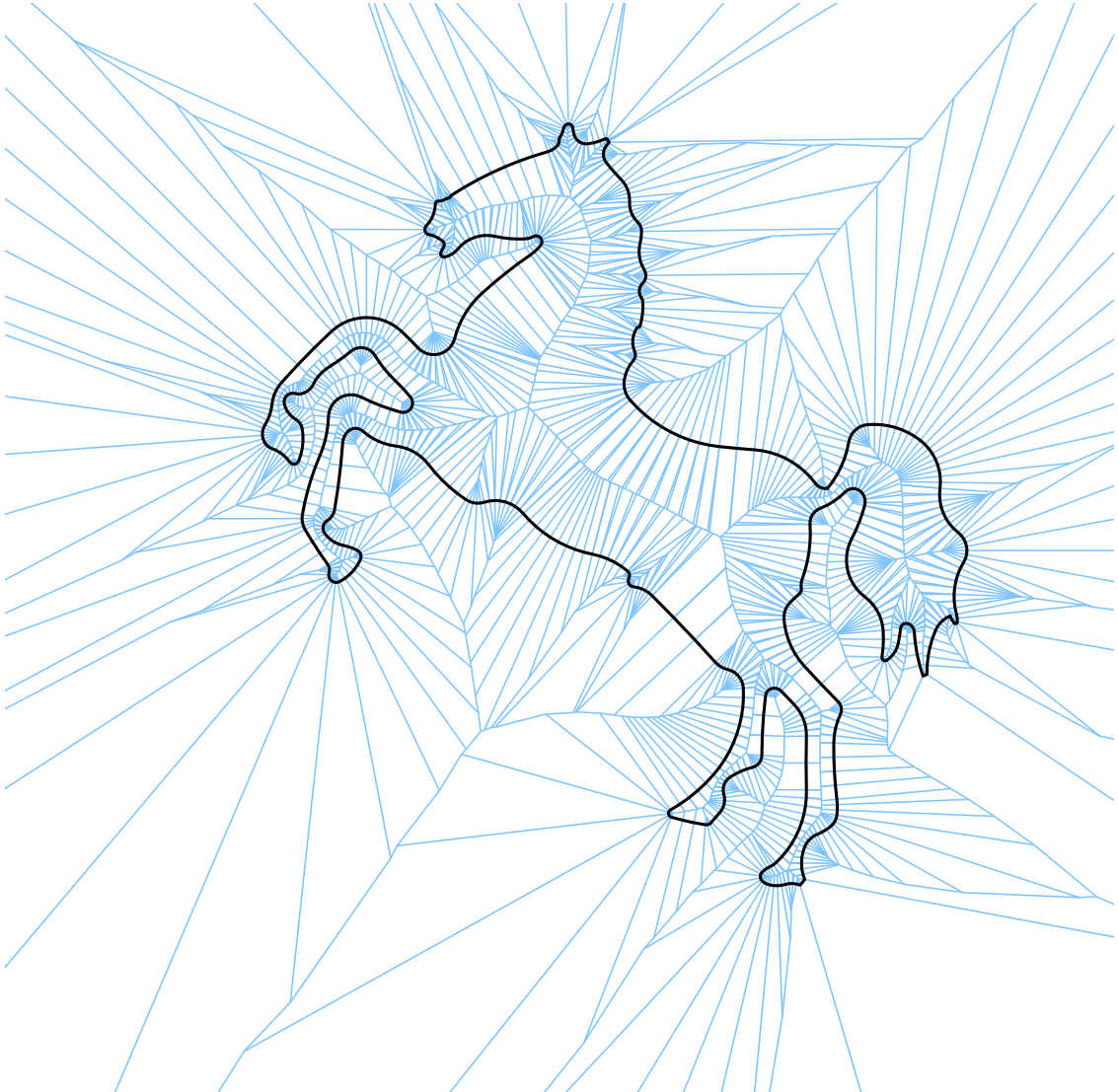
Straight Skeletons

- Aichholzer, Alberts, Aurenhammer, Gärtner 1995.
- Problem: Given input graph, find the *straight skeleton*.



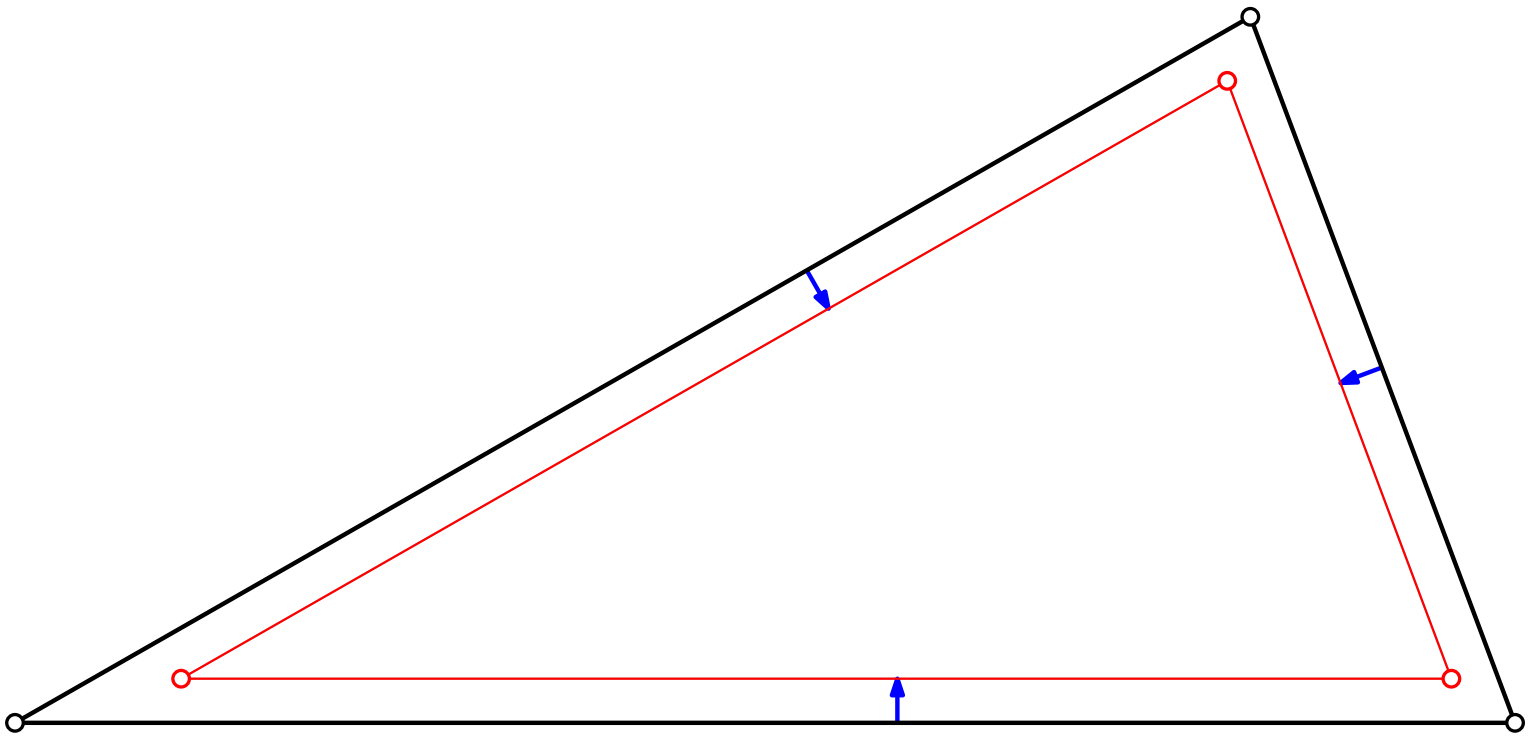
Straight Skeletons

- Aichholzer, Alberts, Aurenhammer, Gärtner 1995.
- Problem: Given input graph, find the *straight skeleton*.



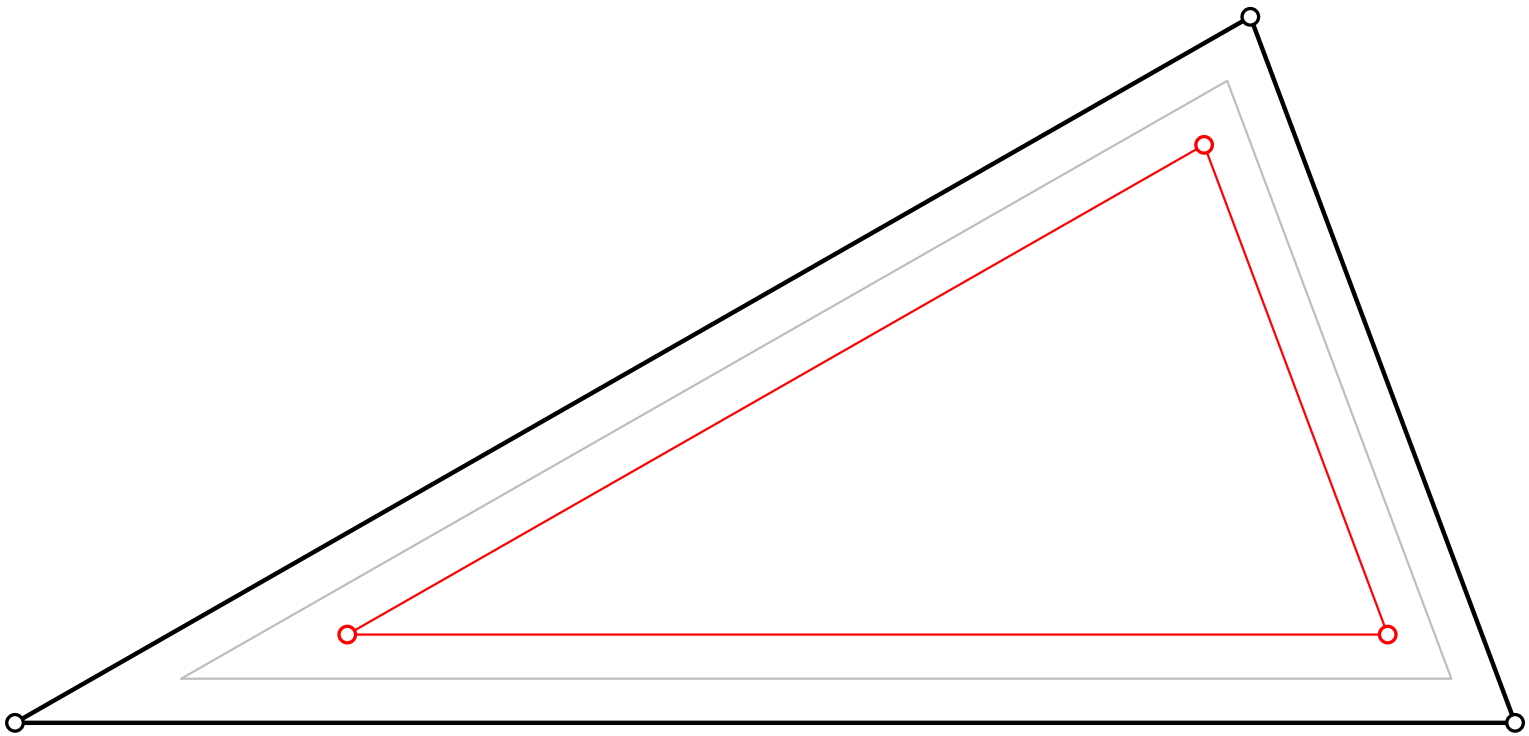
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.



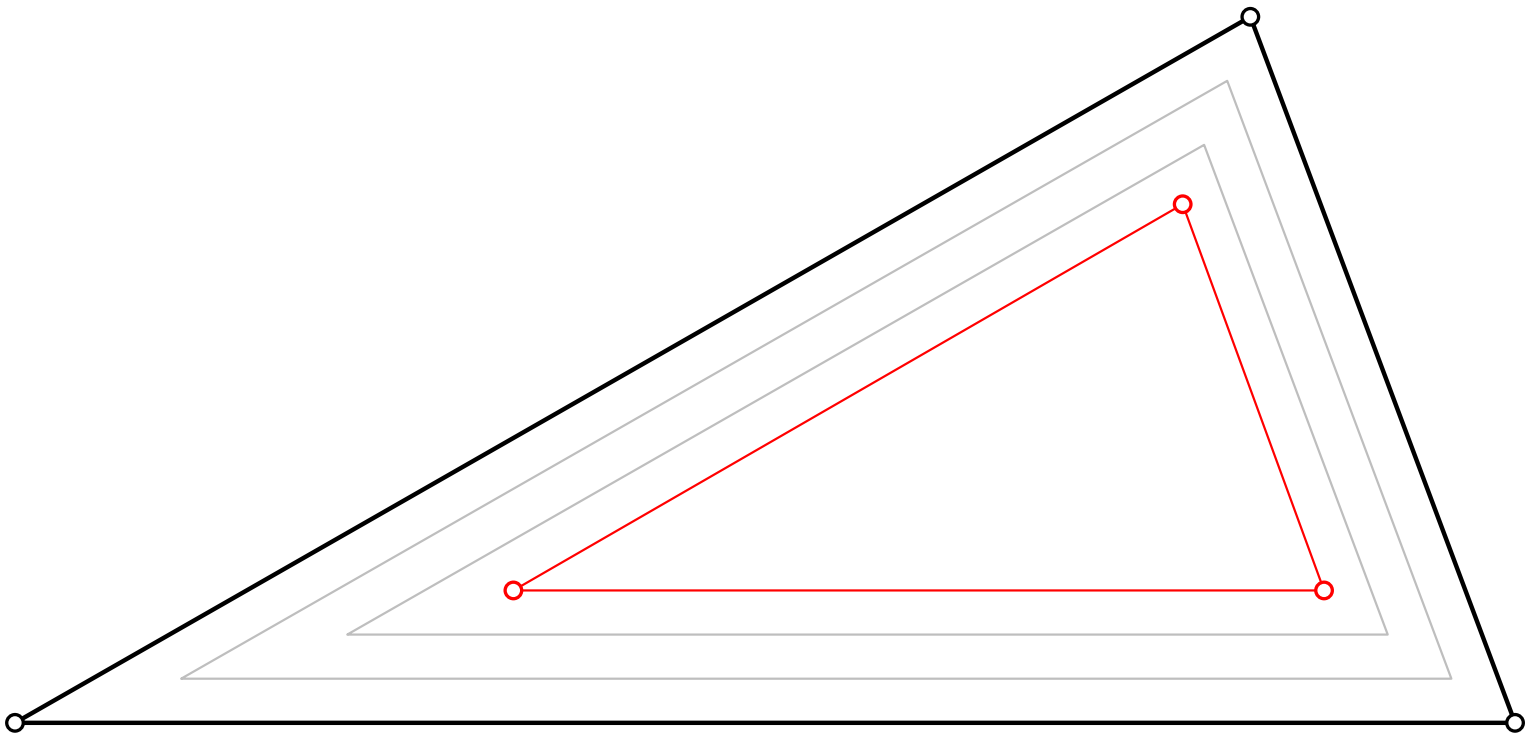
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



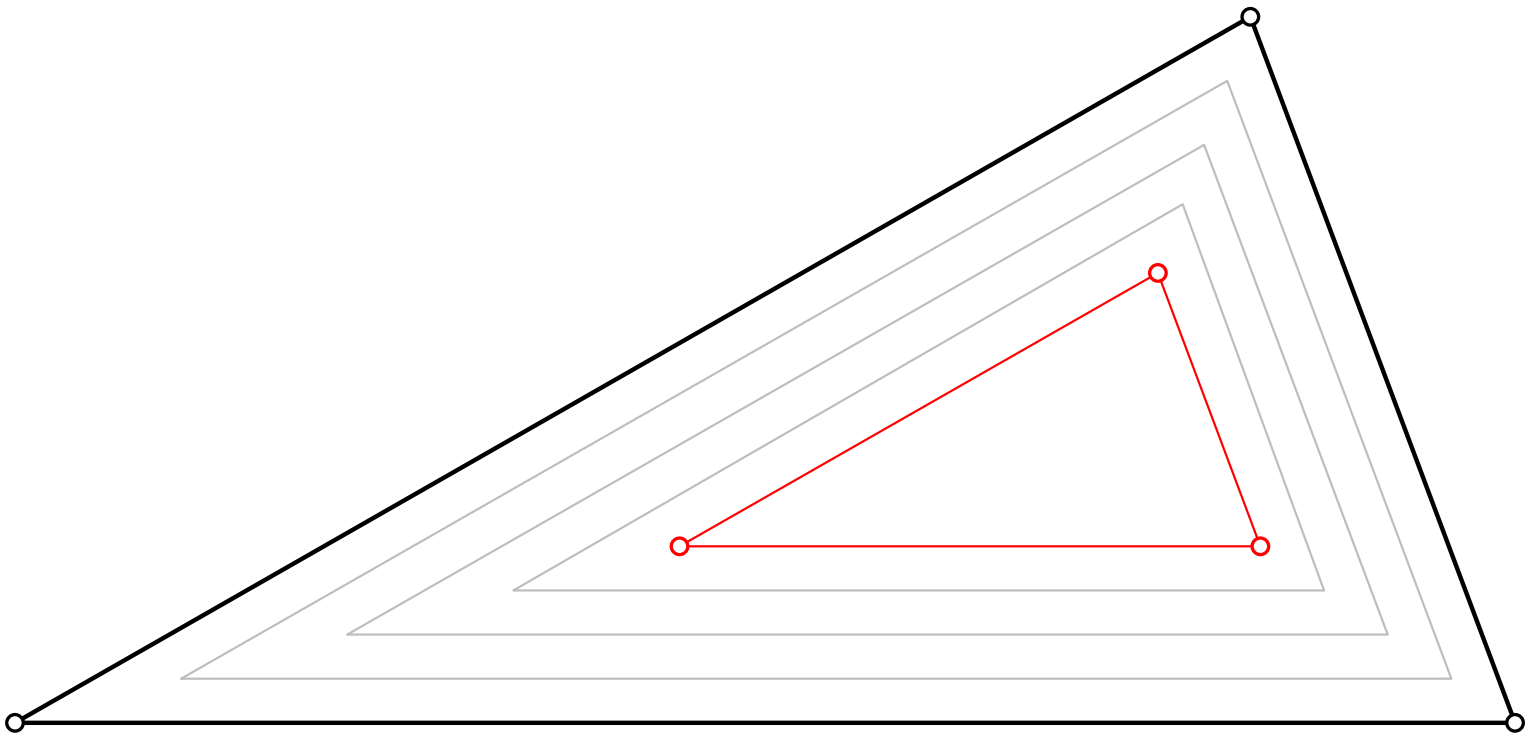
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



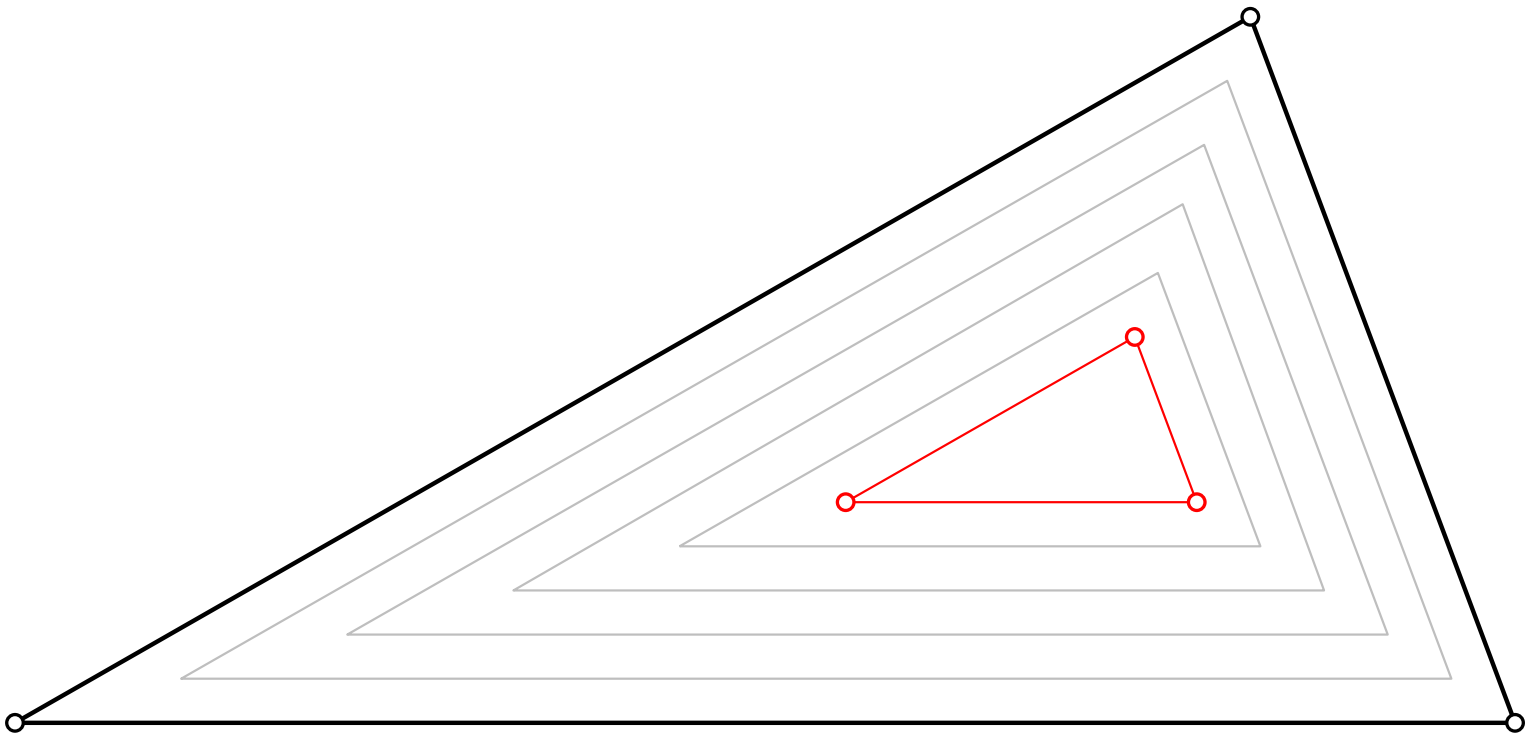
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



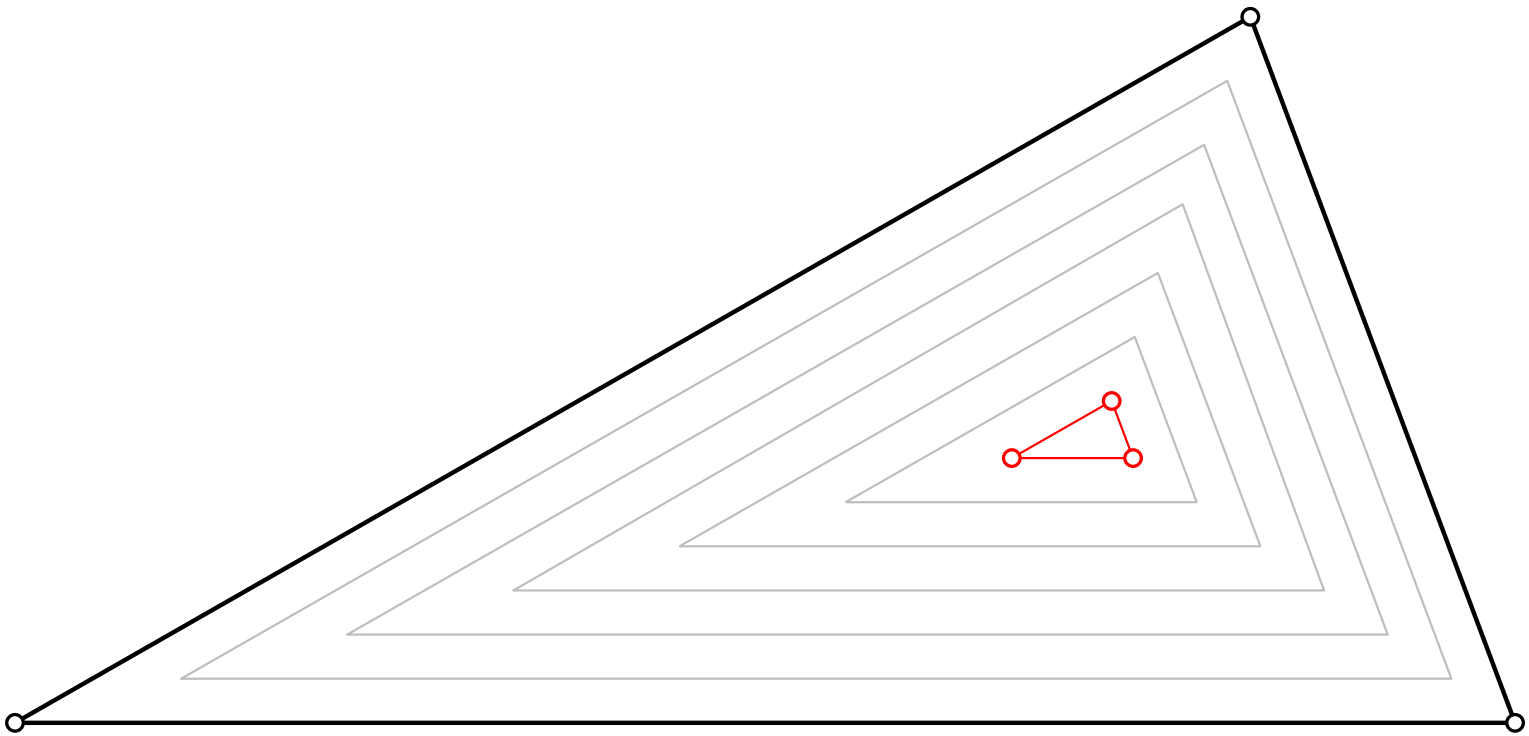
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



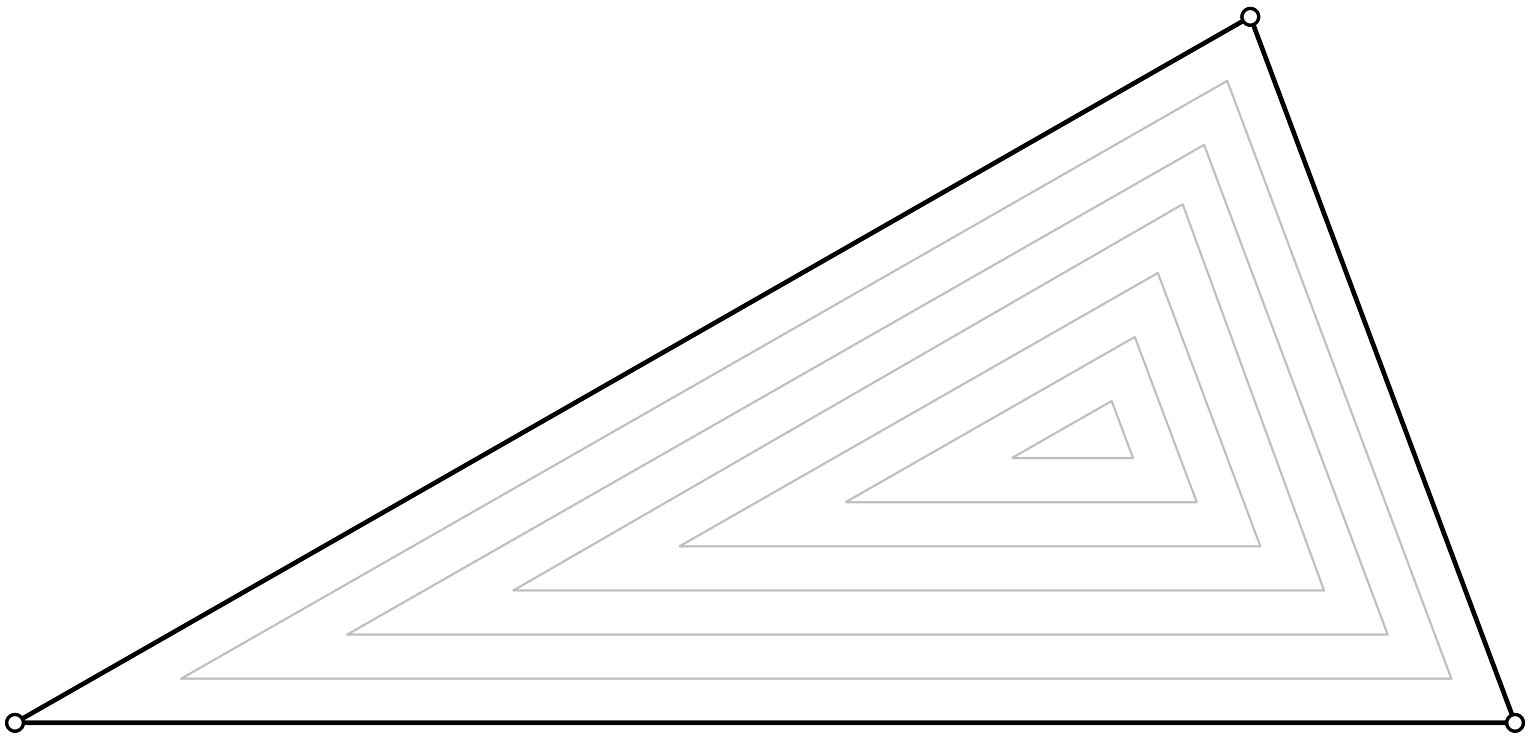
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



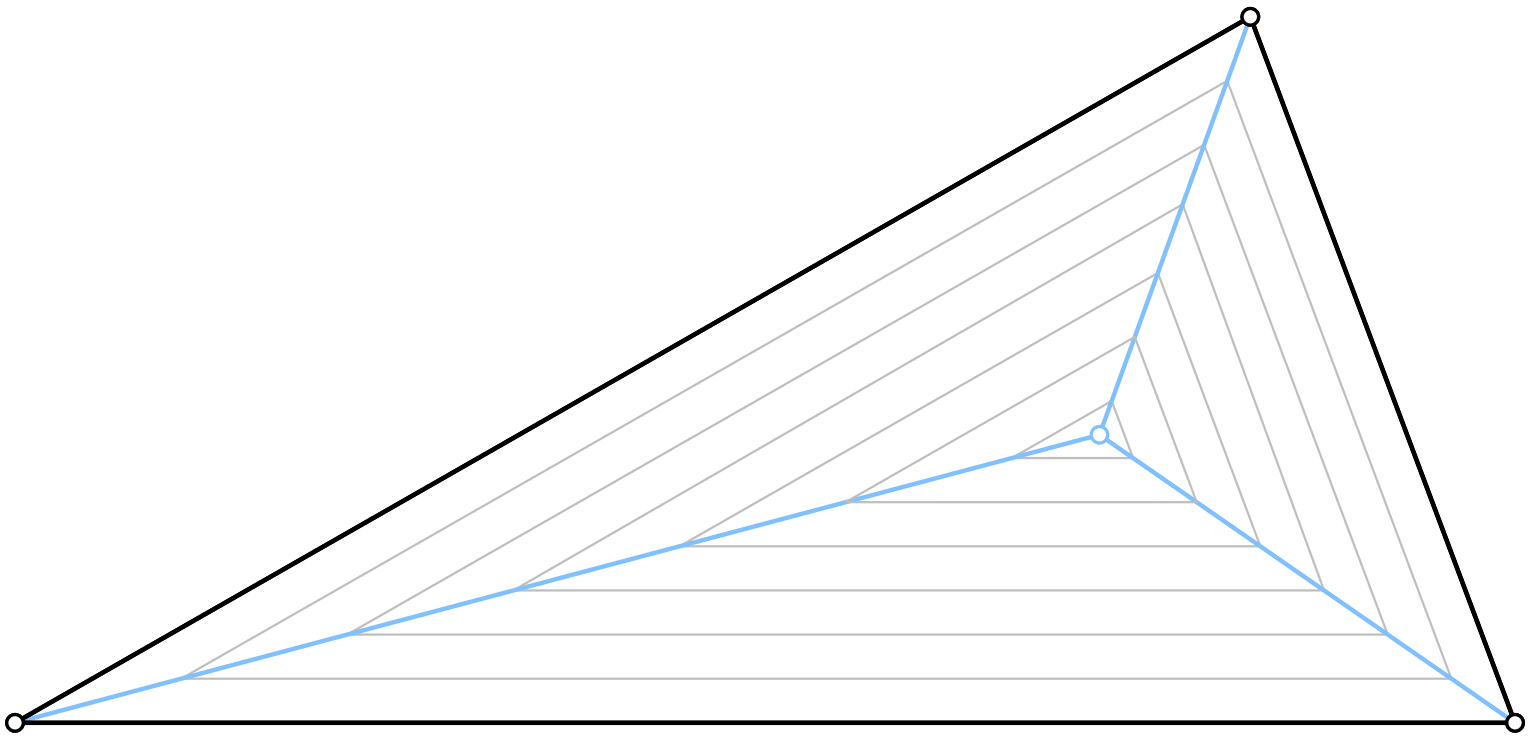
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.



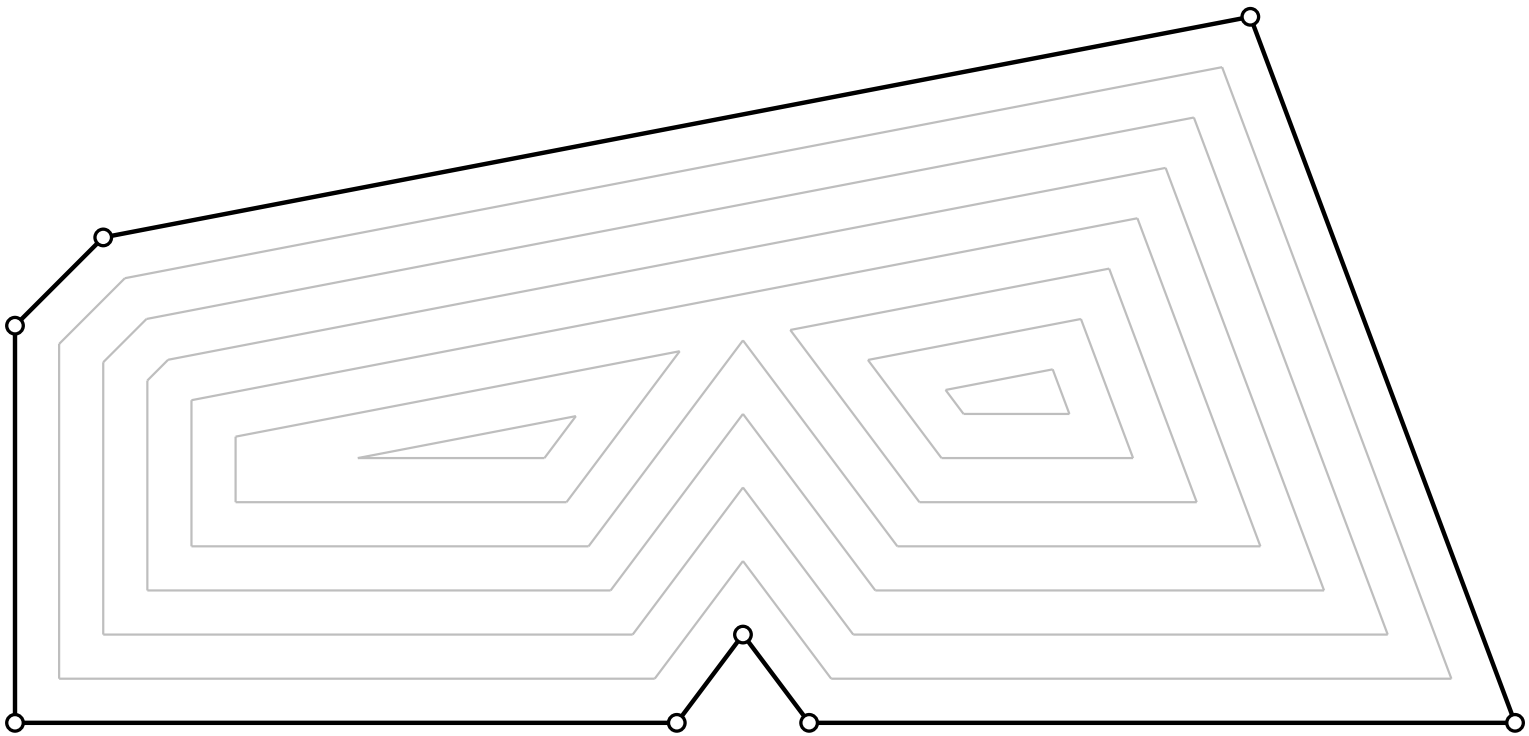
Straight Skeletons – Motivation

- input polygon $\mathcal{P}$ emanates wavefront $\mathcal{WF}(\mathcal{P}, t)$.
- wavefront propagation — shrinking process.
- straight skeleton $\mathcal{SK}(\mathcal{P})$ is traces of wavefront vertices.



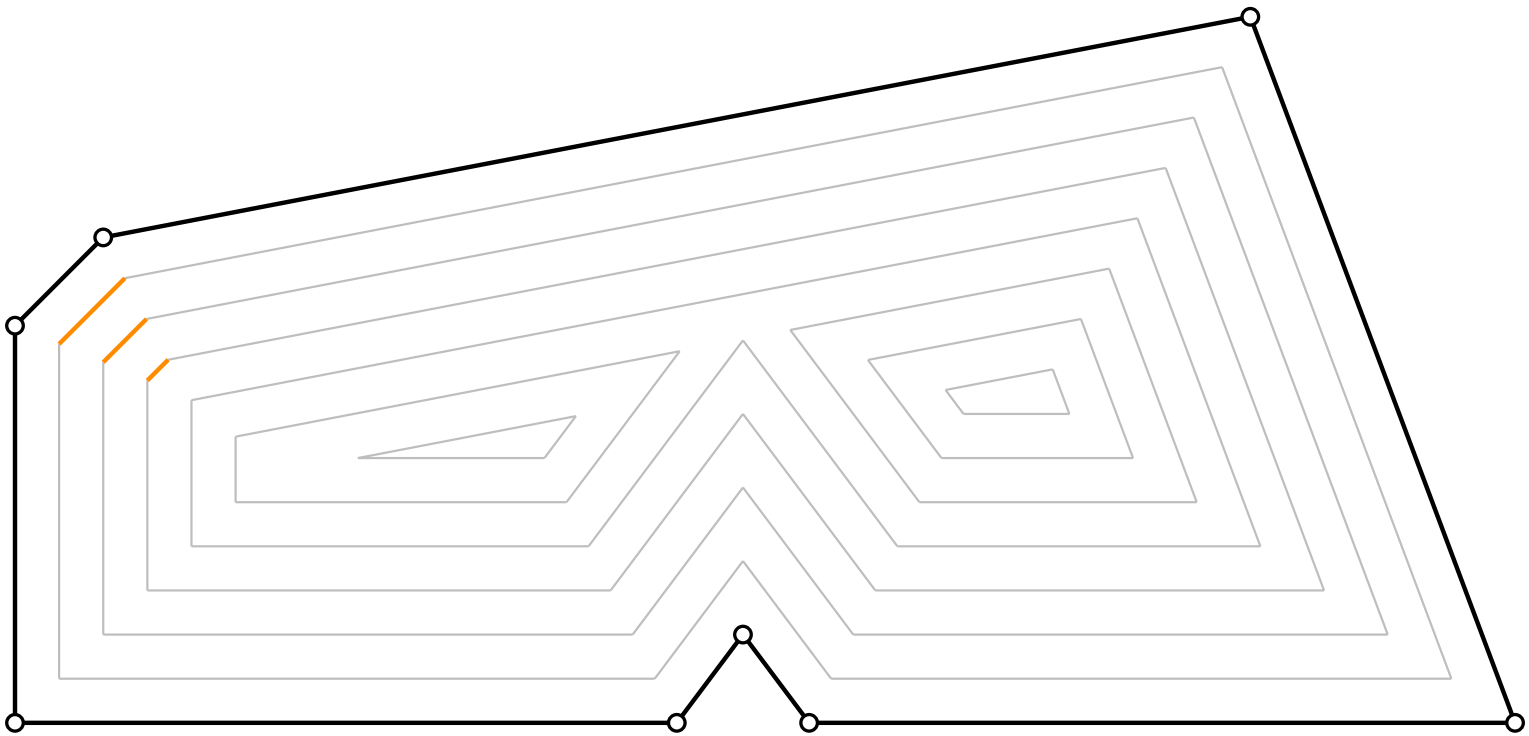
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.



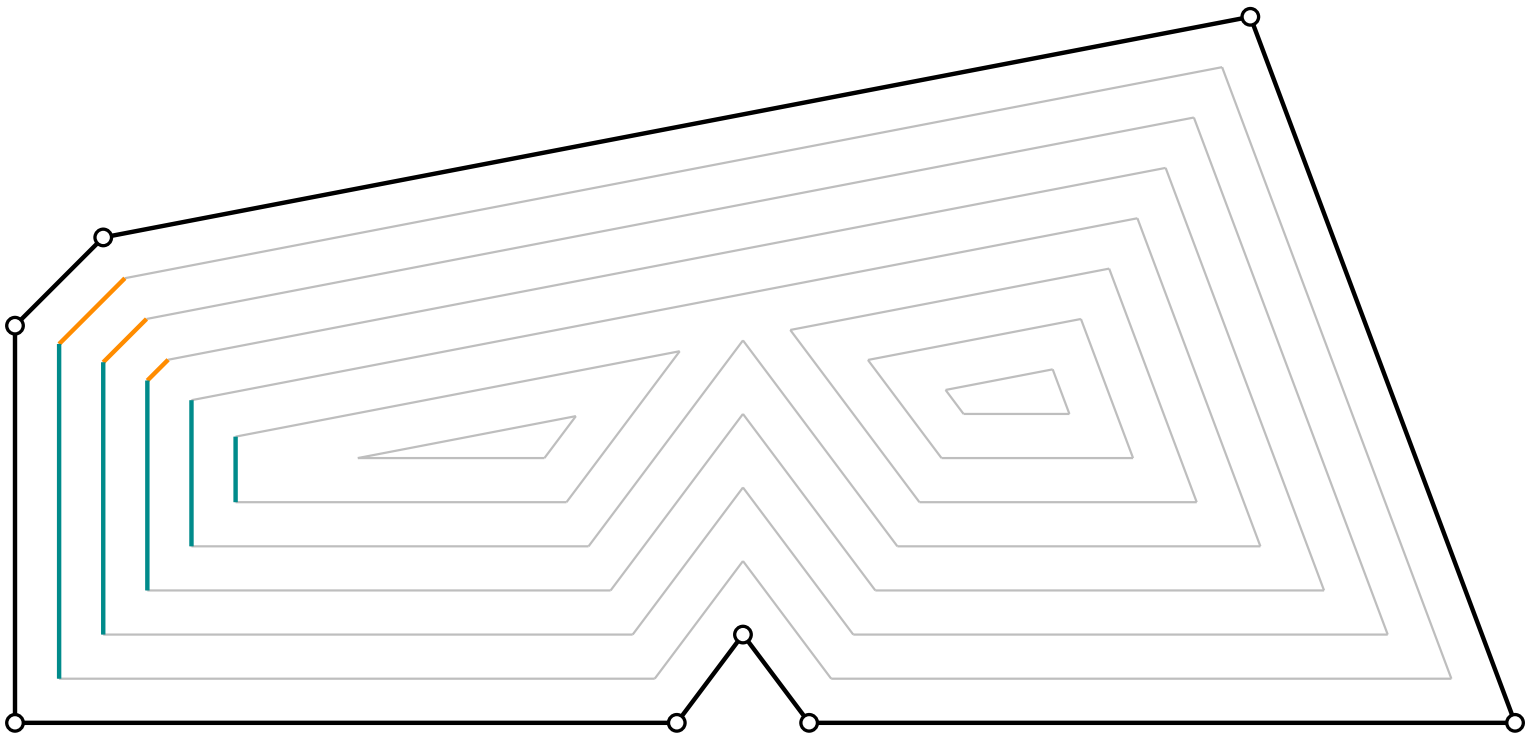
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.



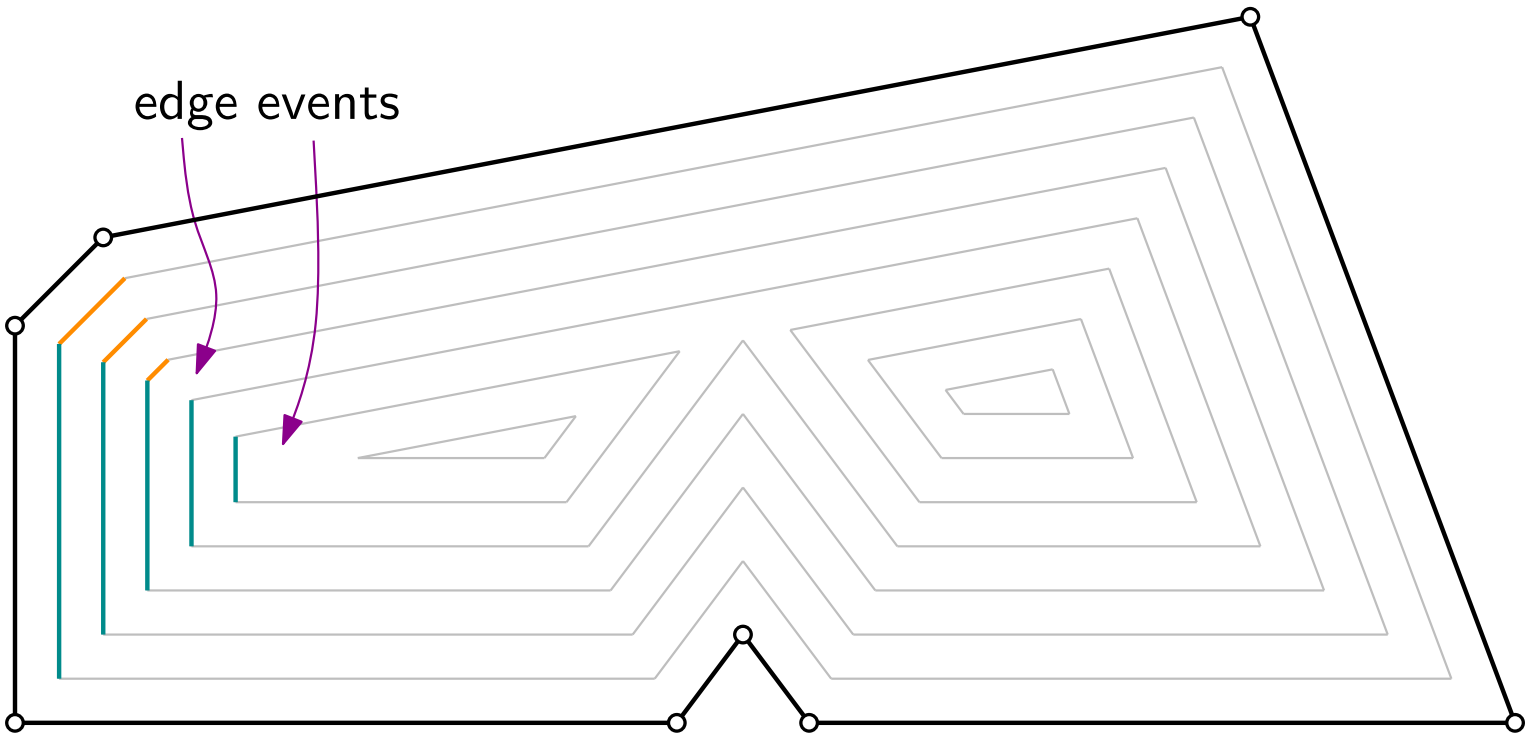
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.



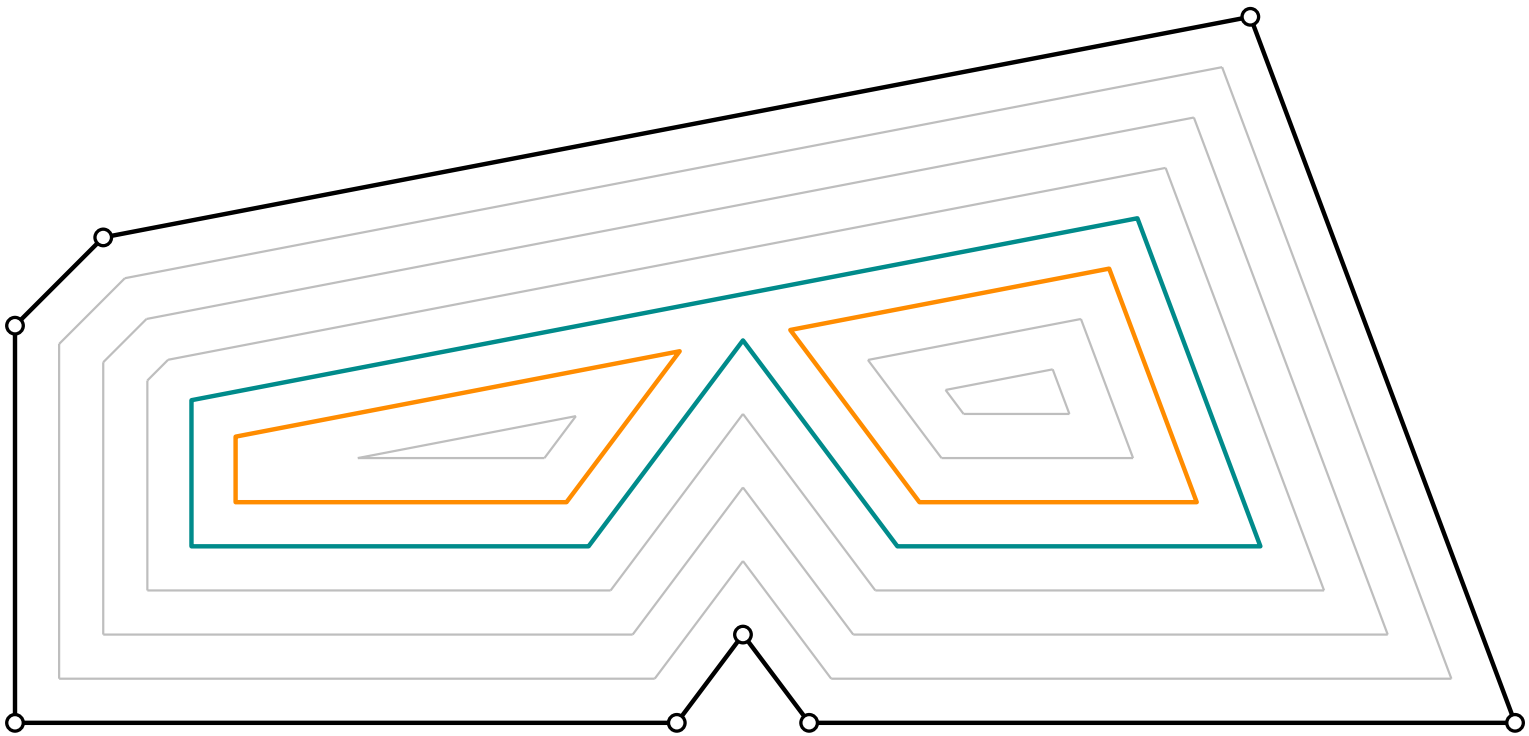
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.
- *edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.



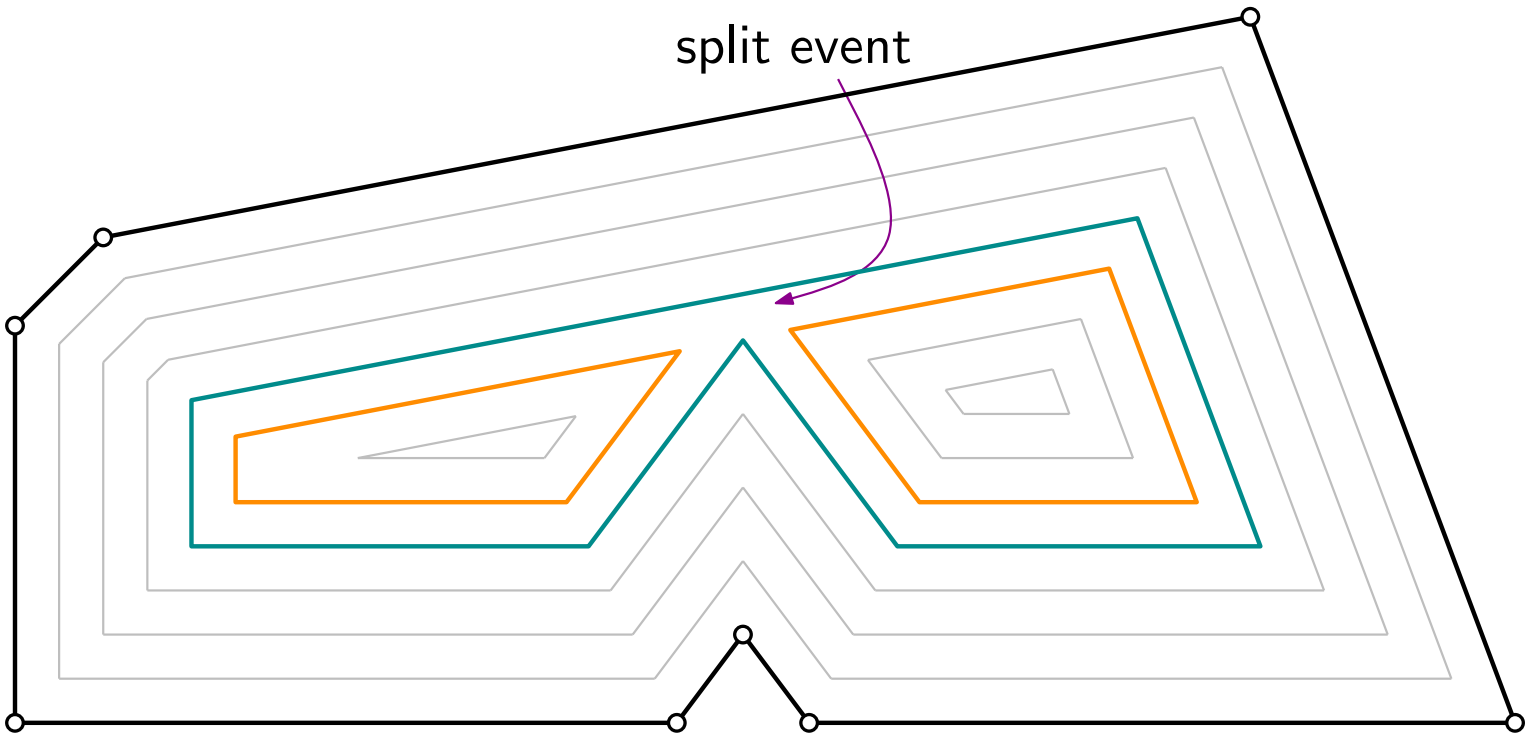
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.
- *edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.



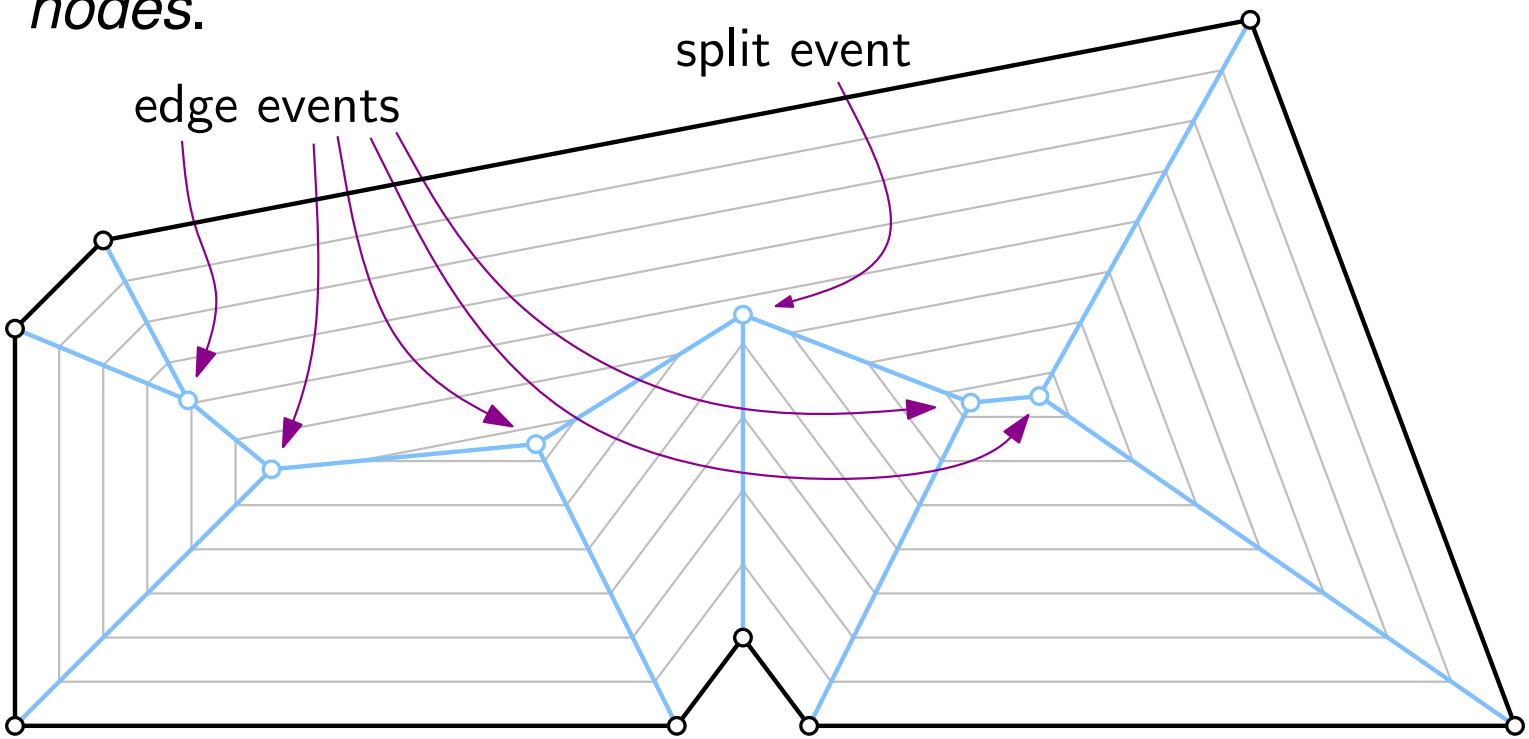
Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.
- *edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.
- *split event*: wavefront splits into two parts.



Topolgy Changes – Edge- and Split-Events

- Wavefront topology changes over time.
- *edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.
- *split event*: wavefront splits into two parts.
- In $\mathcal{SK}(\mathcal{P})$, events (topology changes) are witnessed by *nodes*.



Applications: Roof Modeling

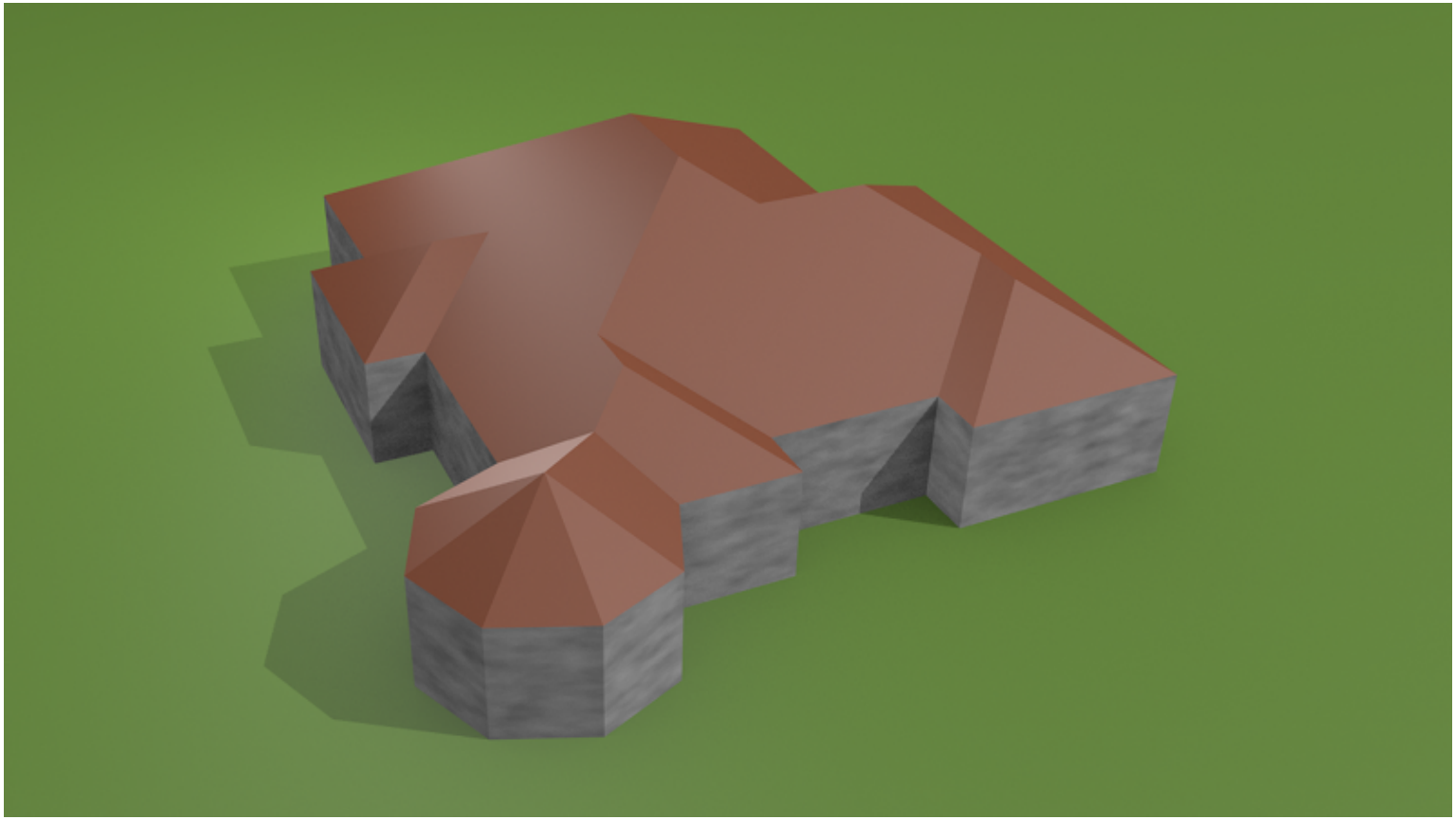
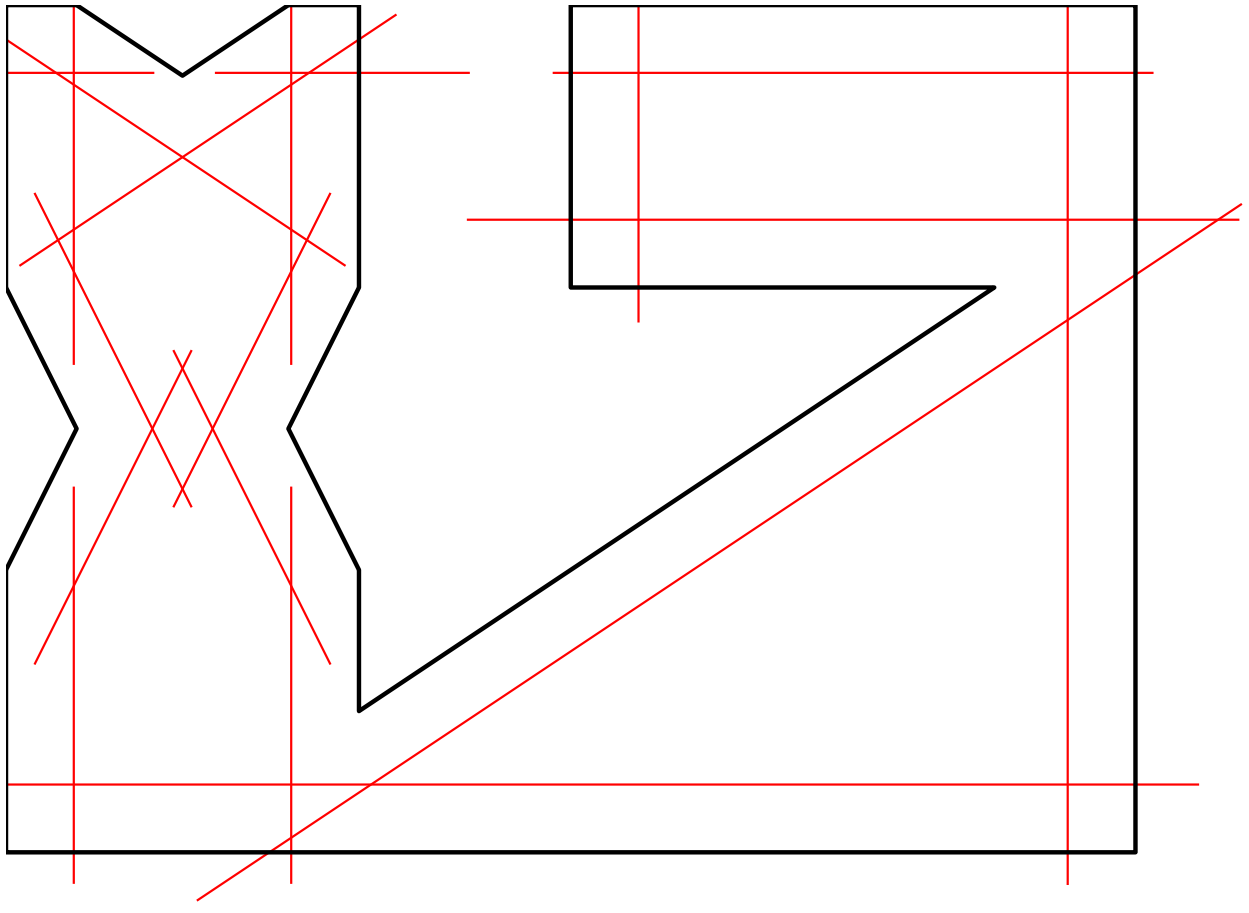
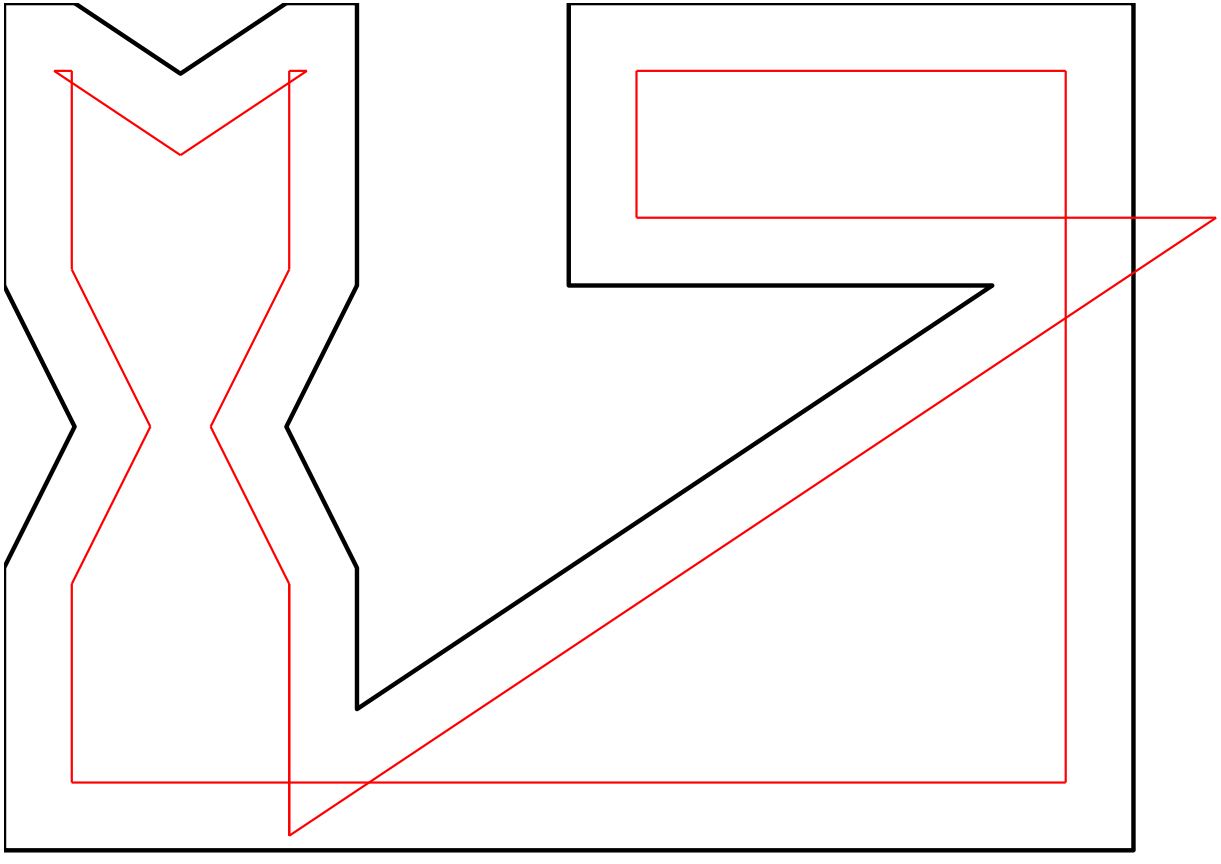


image credit: Stefan Huber

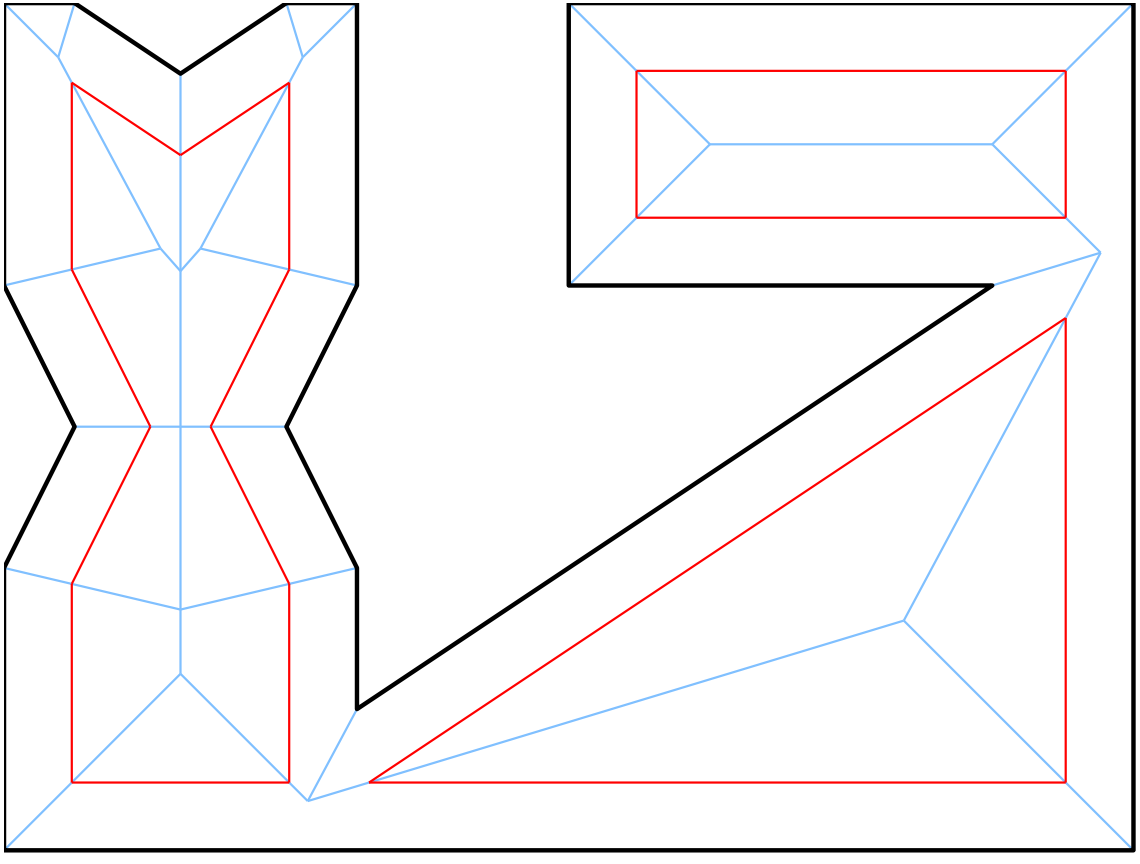
Applications: Offsetting



Applications: Offsetting



Applications: Offsetting

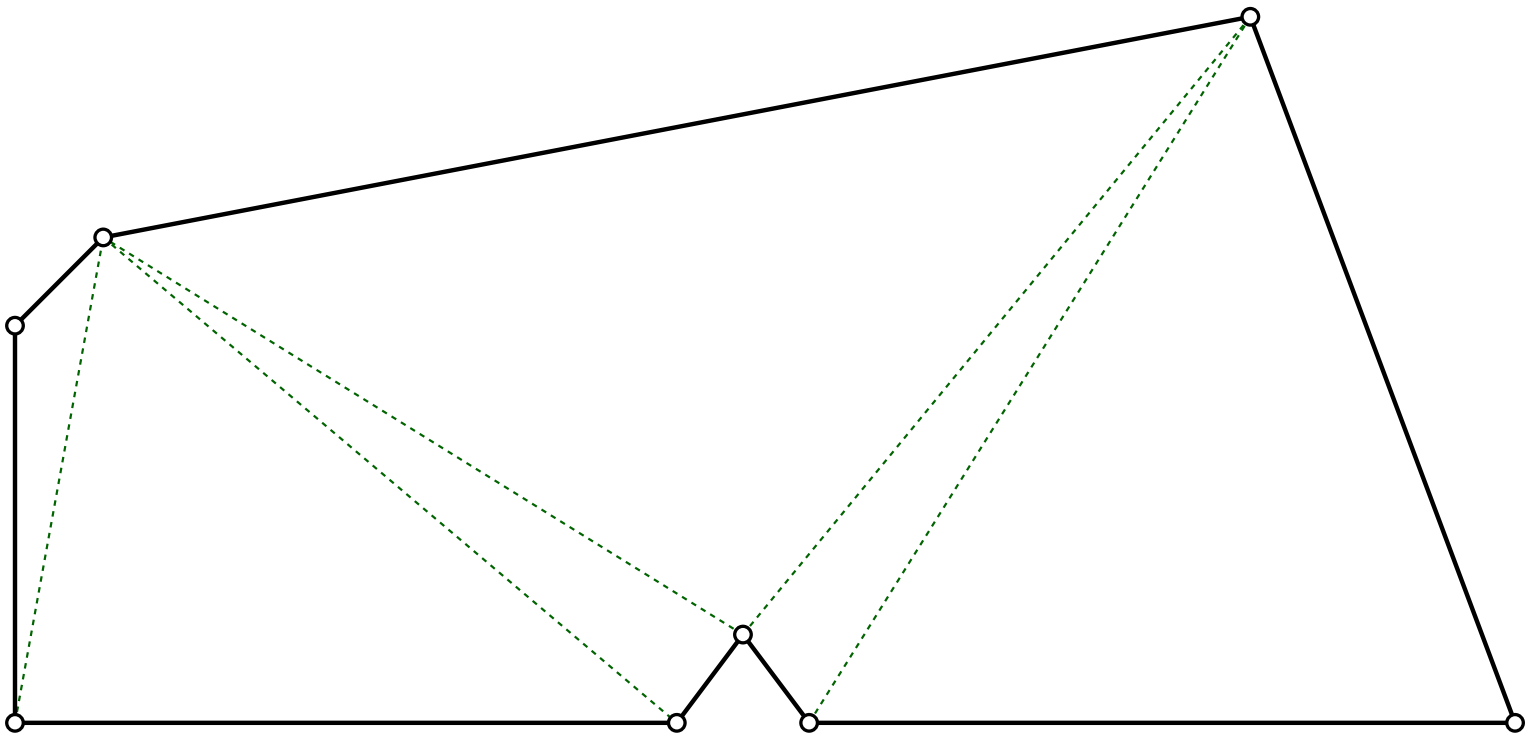


Computing the Straight Skeleton

- Common approach: simulate the wavefront propagation.
- Problem: When will the next event happen, and what is it?
- If we solve this, we can incrementally construct the SK.

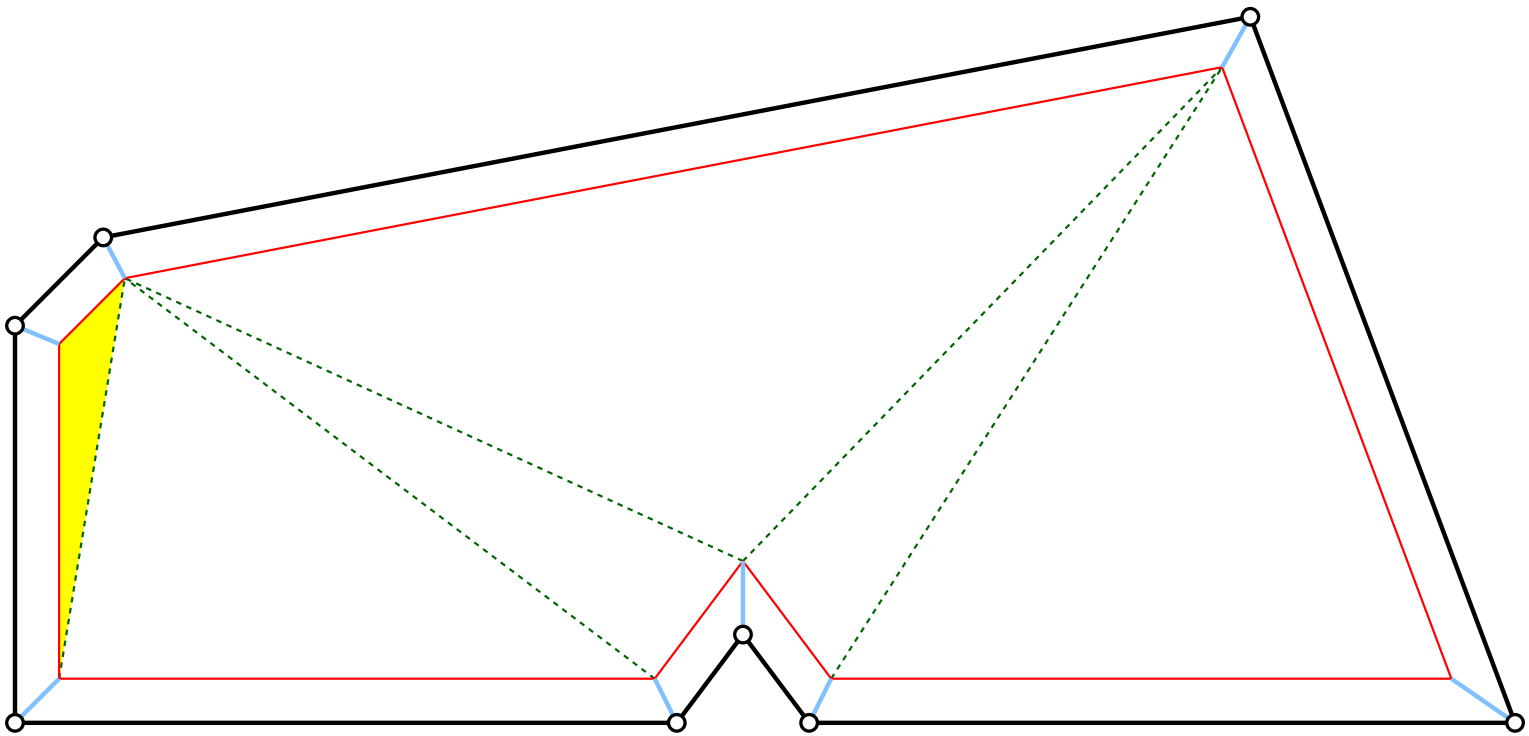
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.



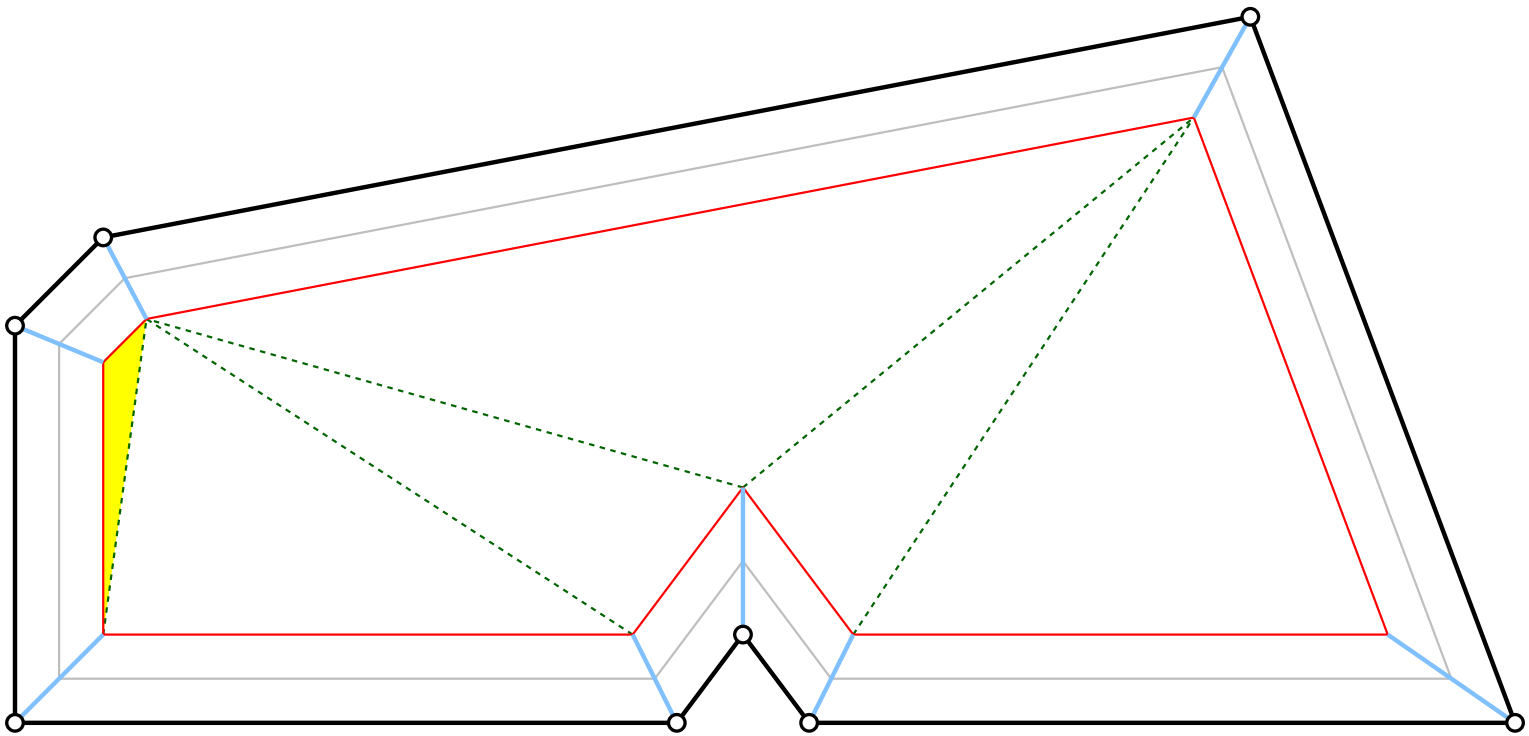
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.



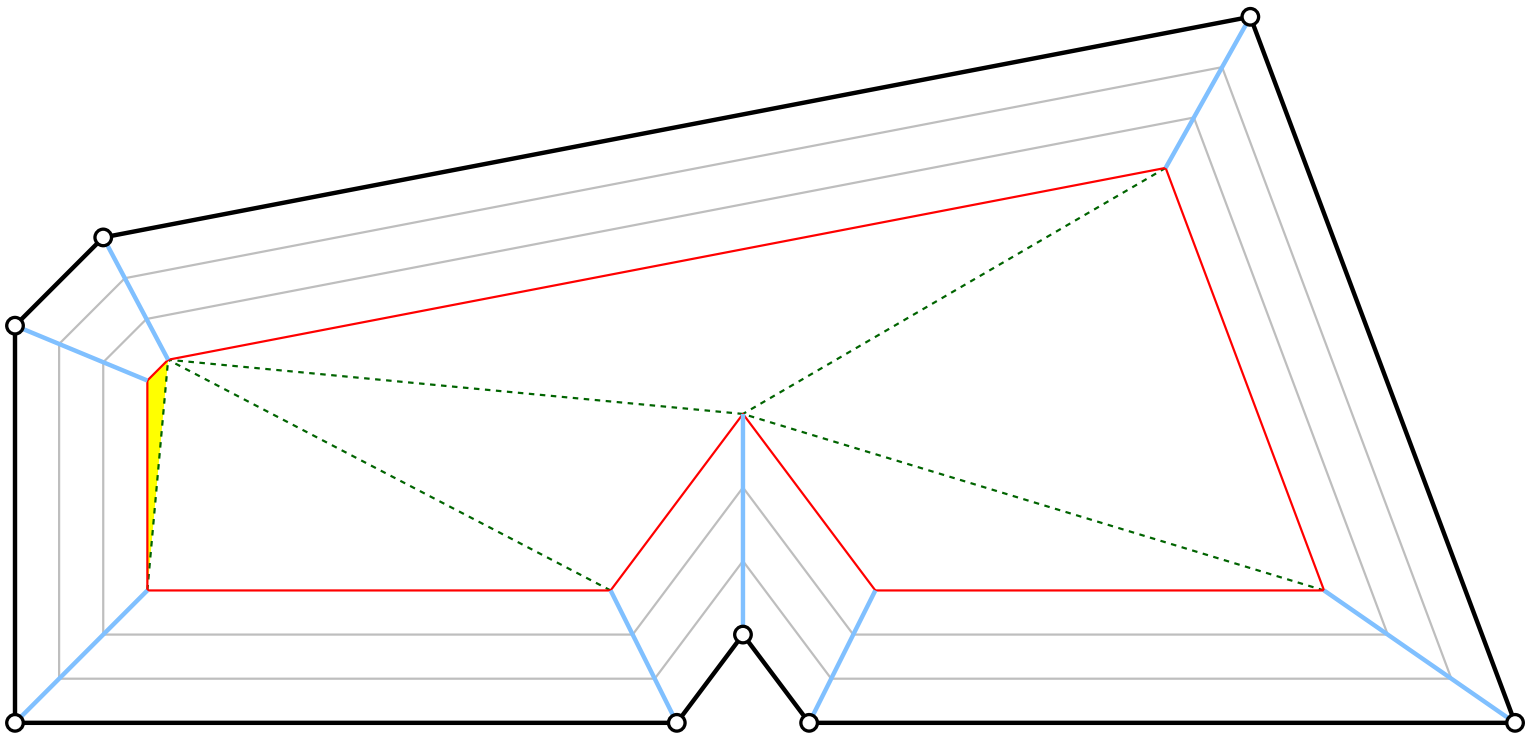
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.



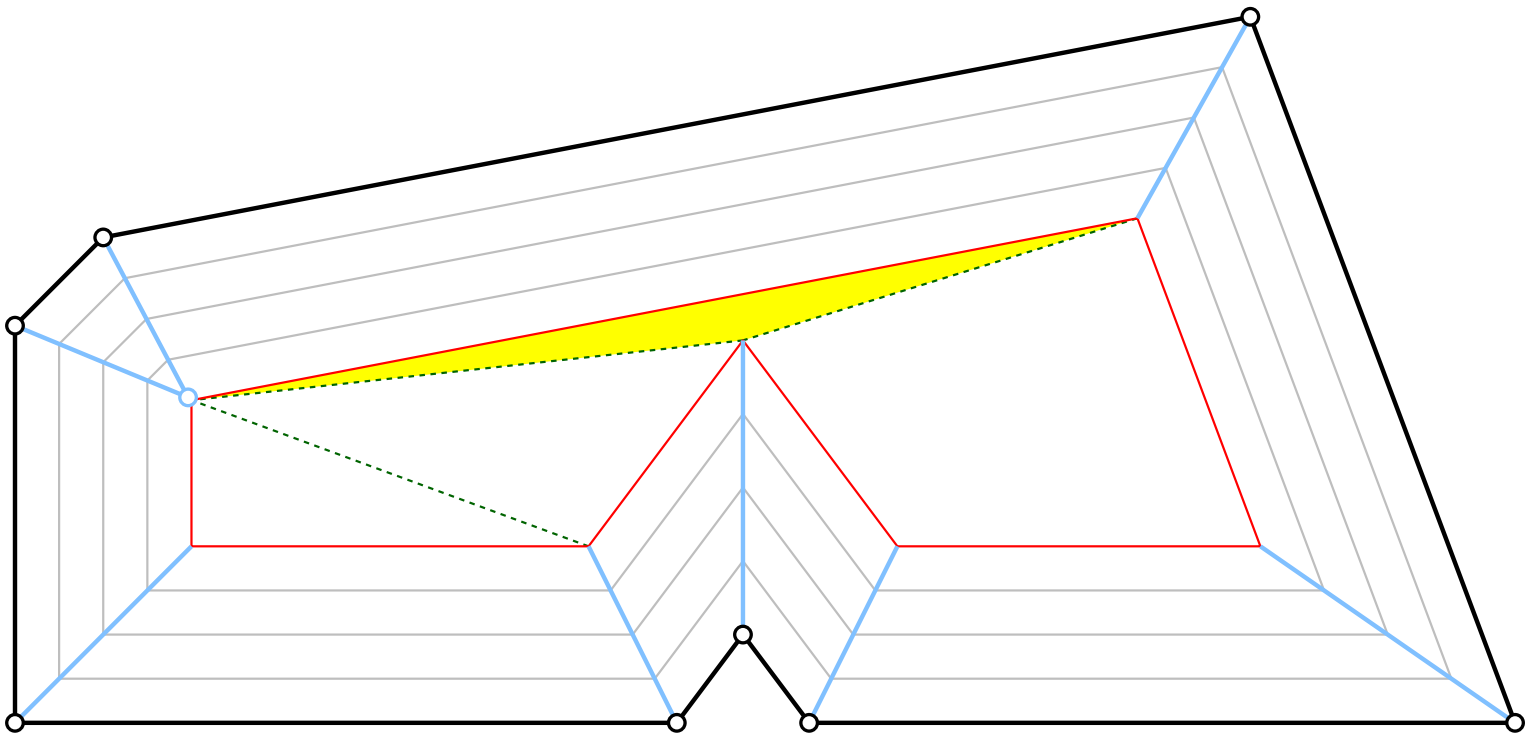
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.



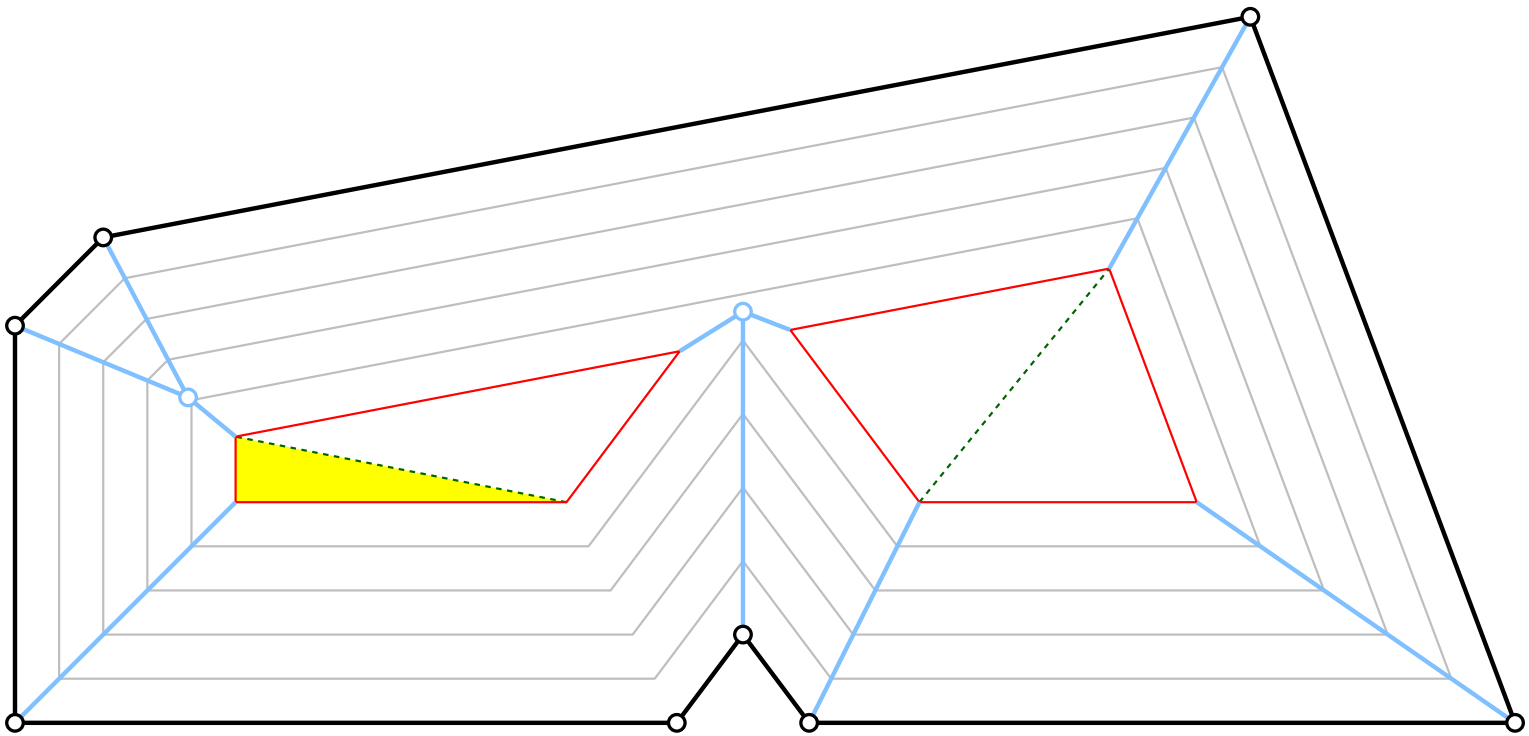
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.



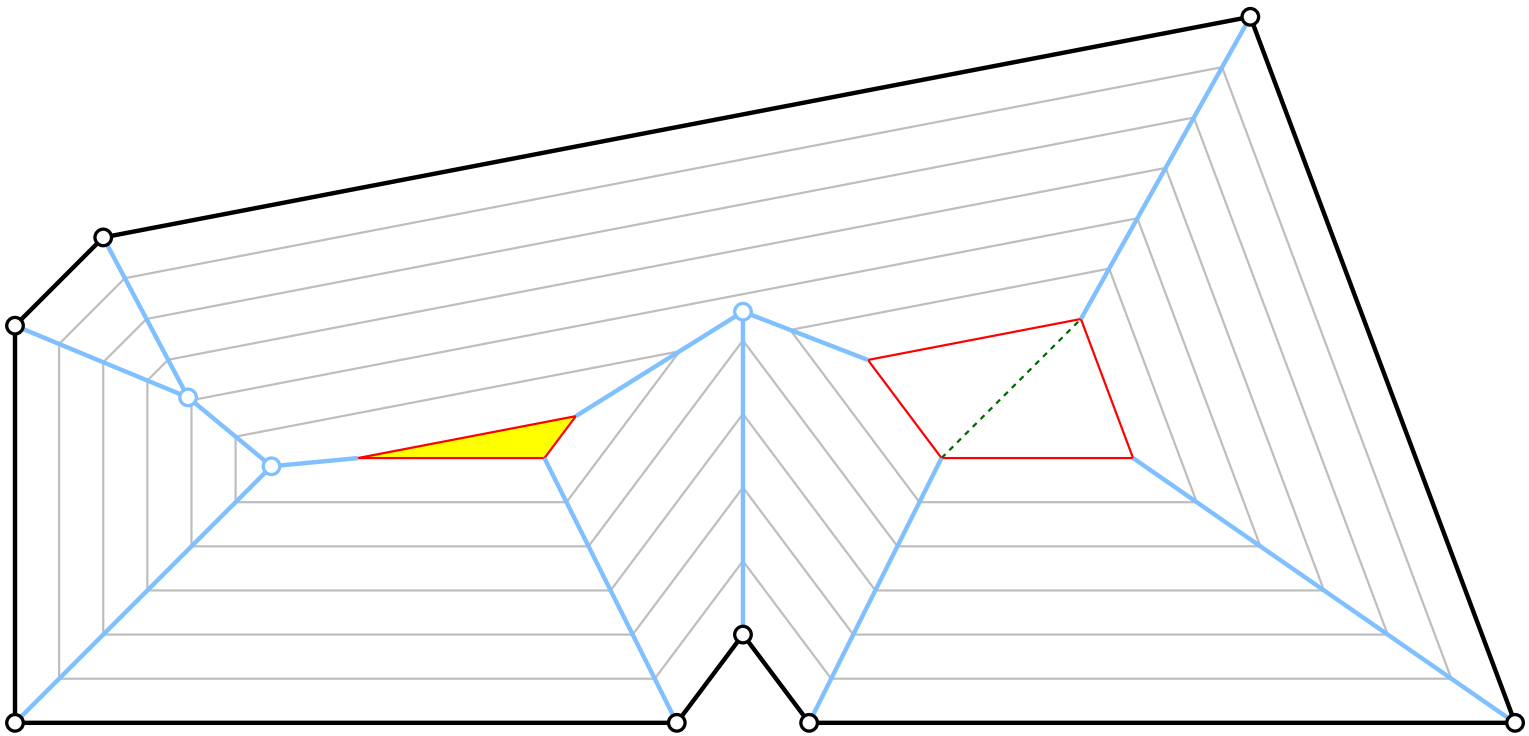
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.



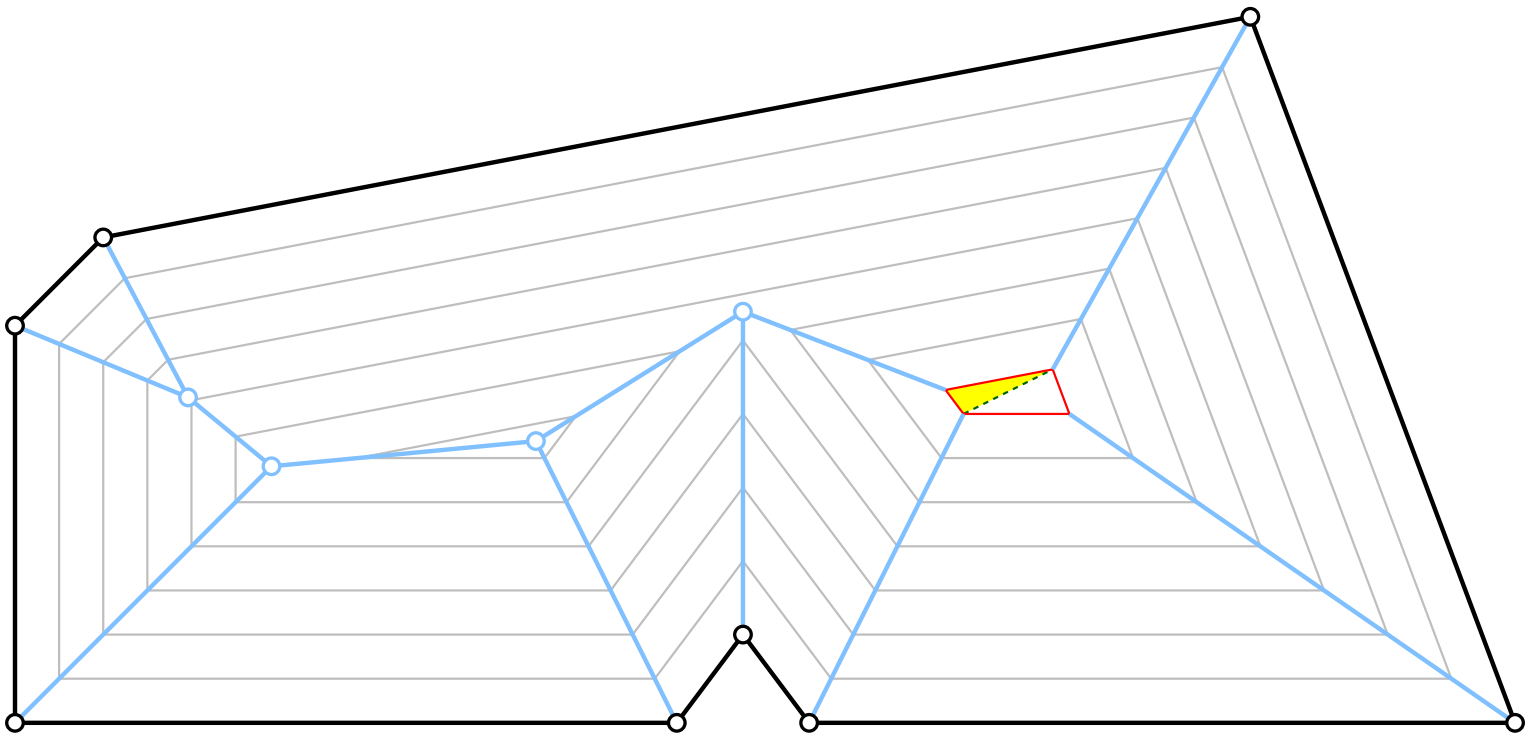
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.



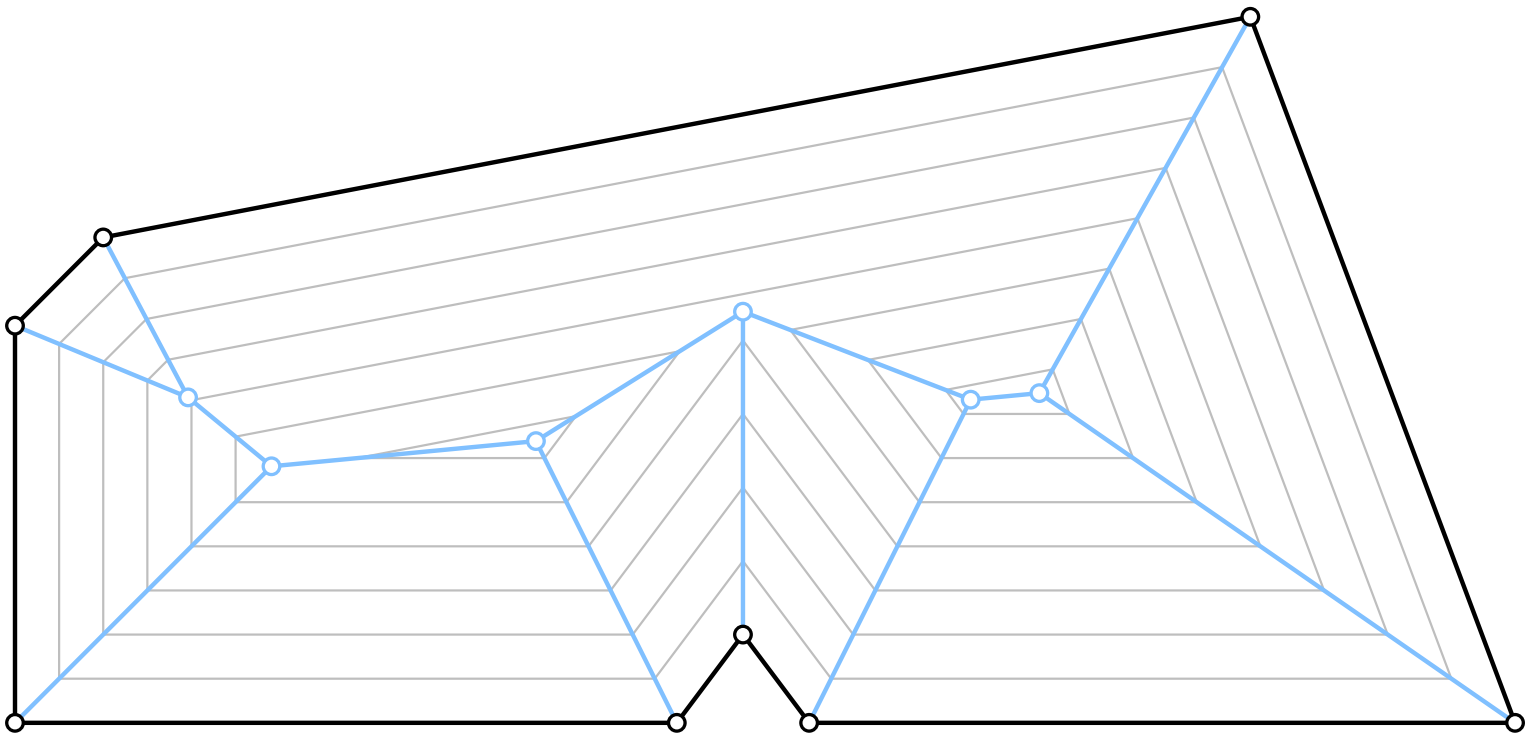
Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.



Triangulation-based Algorithm

- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.



Triangulation-based Algorithm

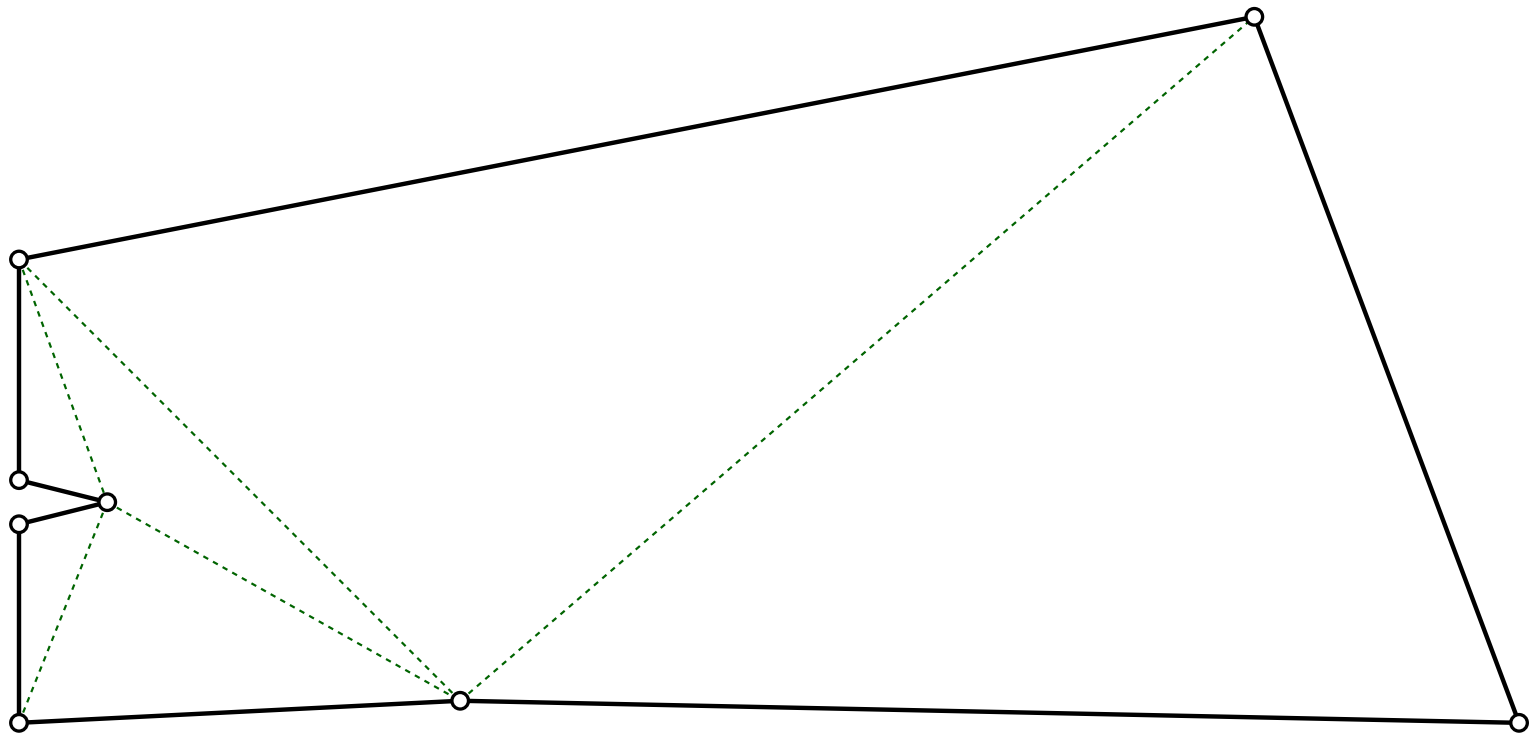
- Aichholzer, Aurenhammer 1996, 1998.
- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.

- Compute collapse times of triangles.
- Maintain a priority queue of collapses.
- On events, update triangulation and priority queue as required.

- We can always easily find the next event, and thus compute the straight skeleton.

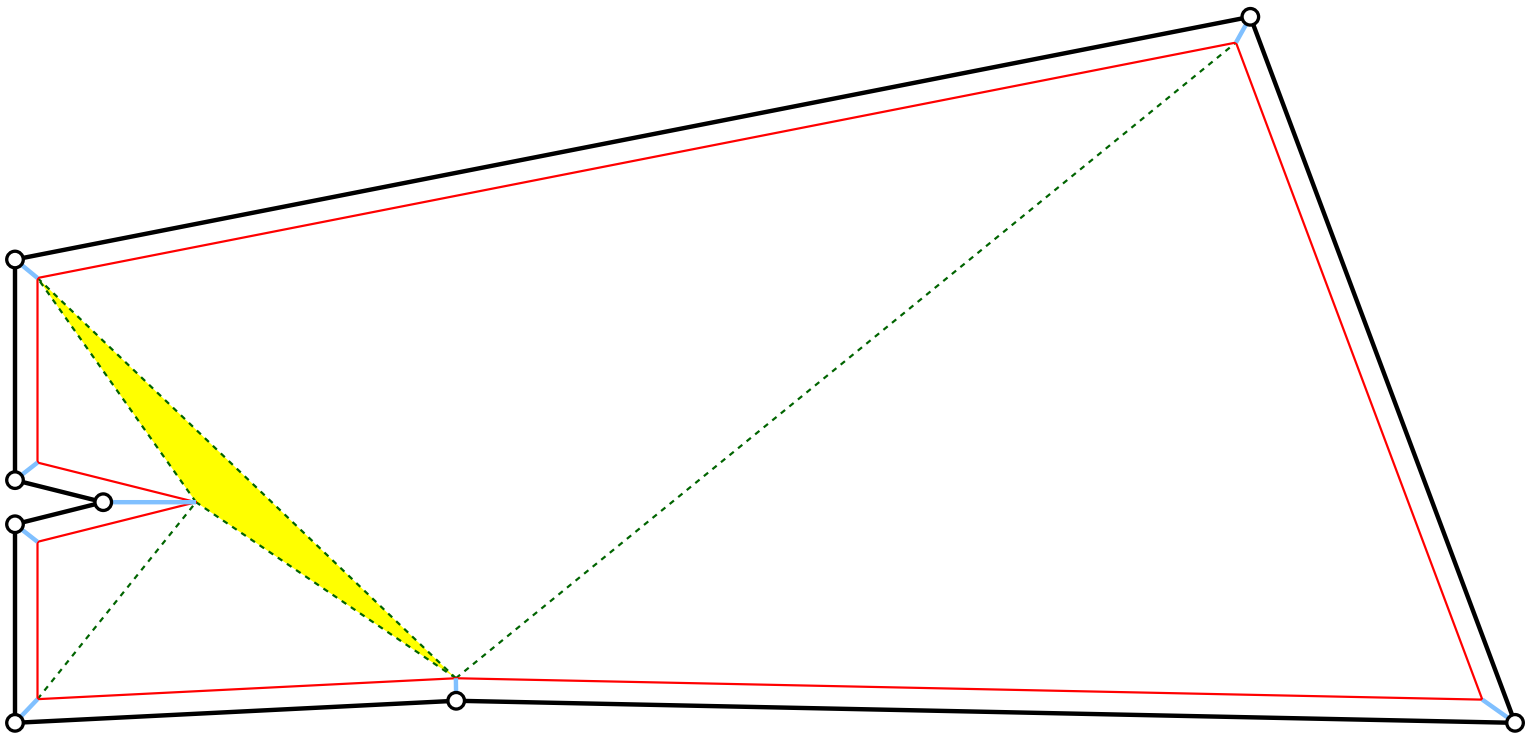
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.



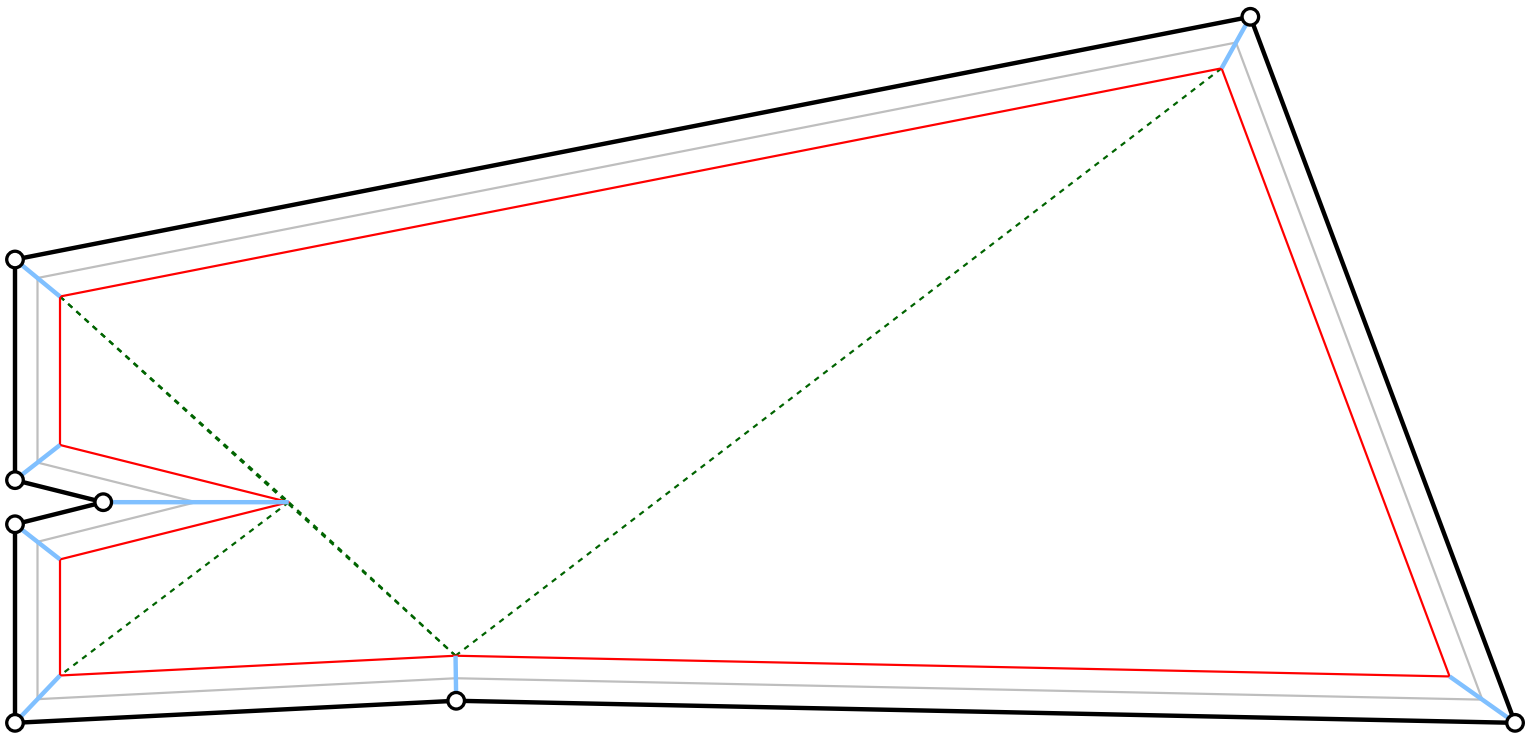
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.



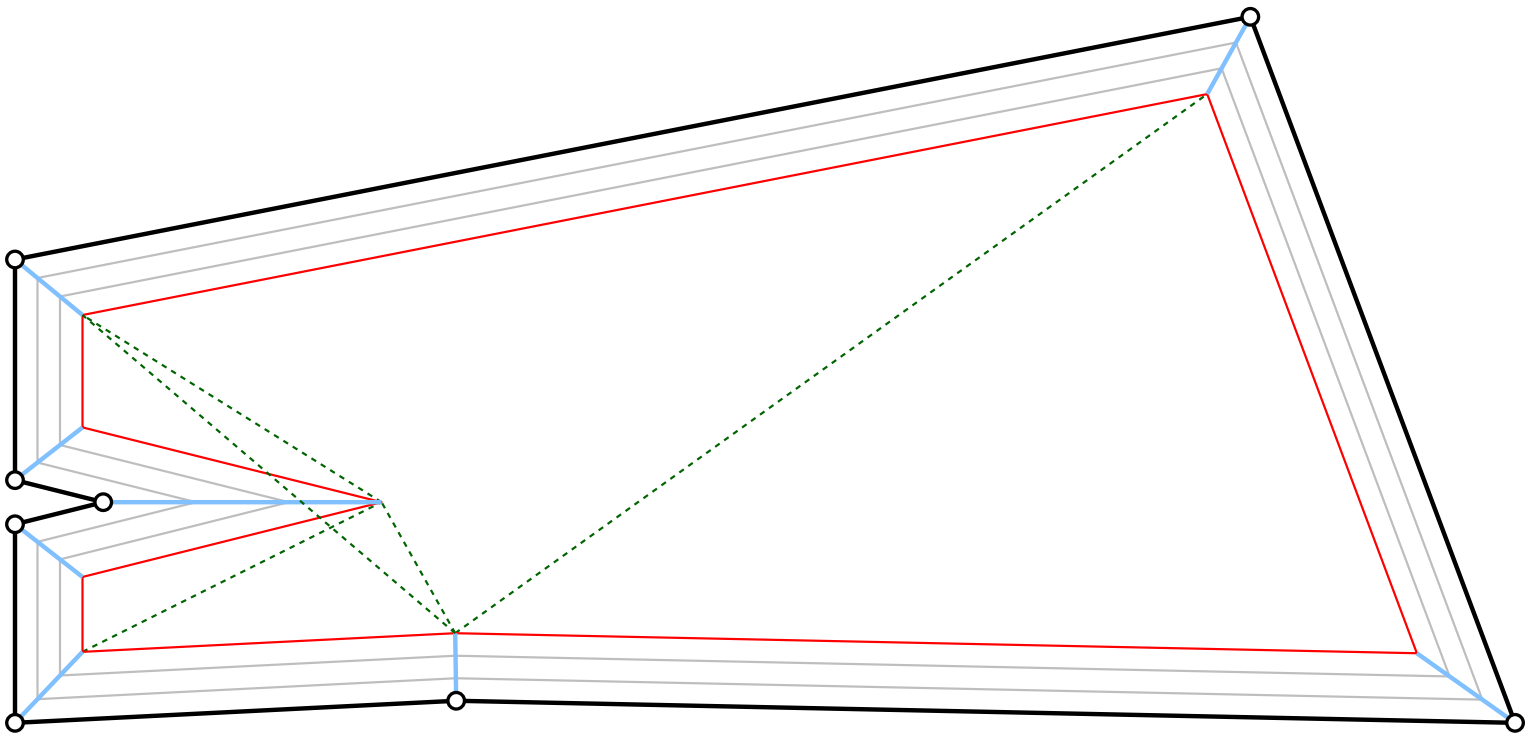
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.



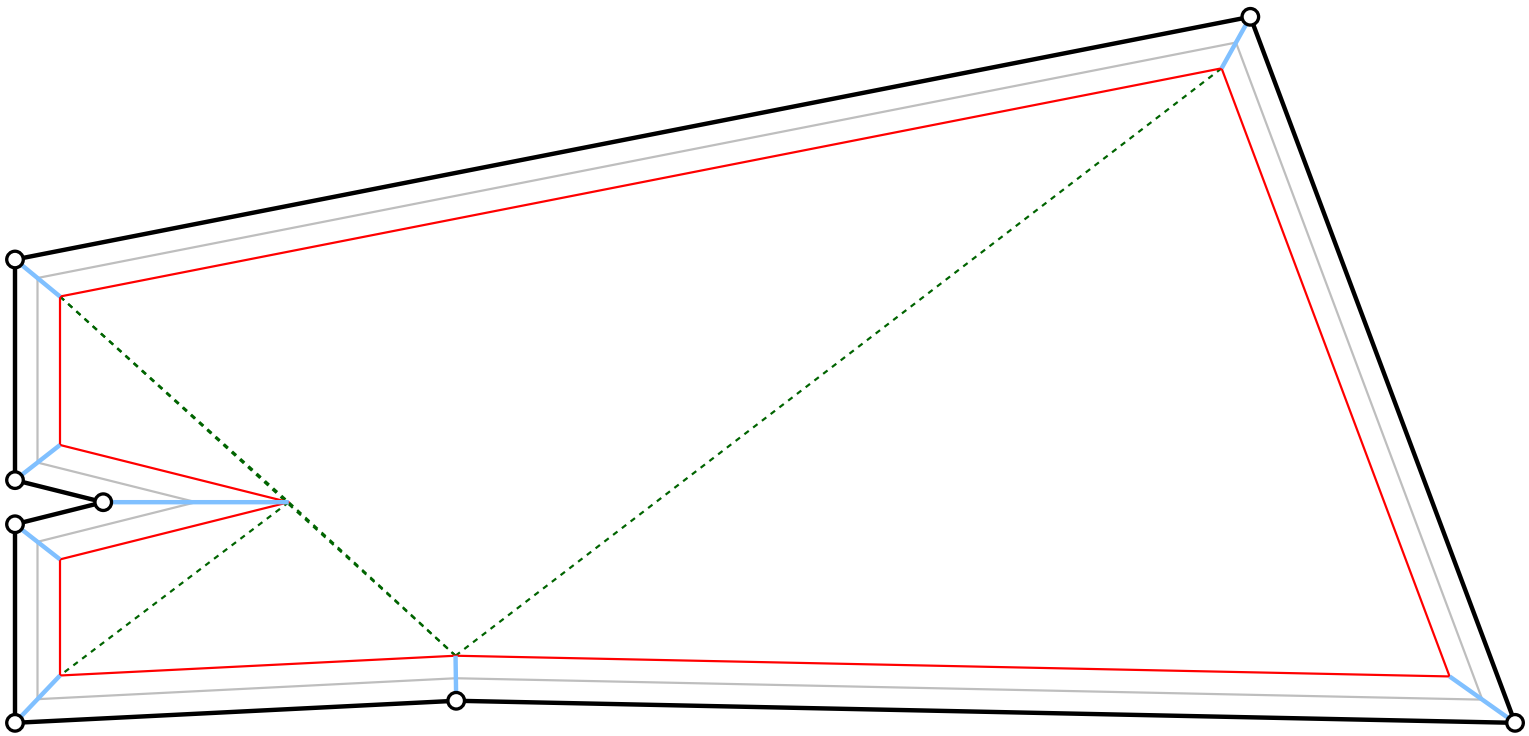
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.



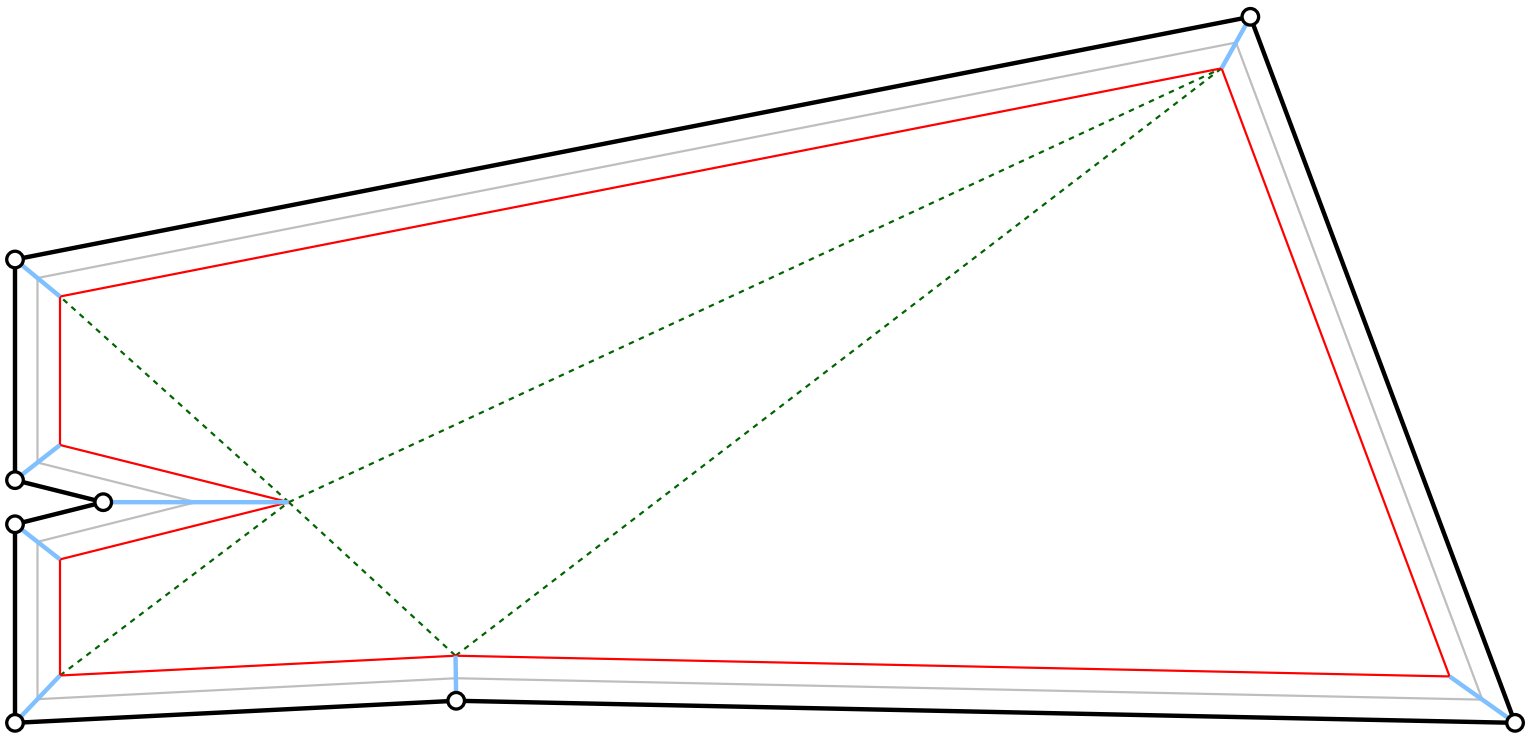
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.
- Instead they need special processing: *flip events*.



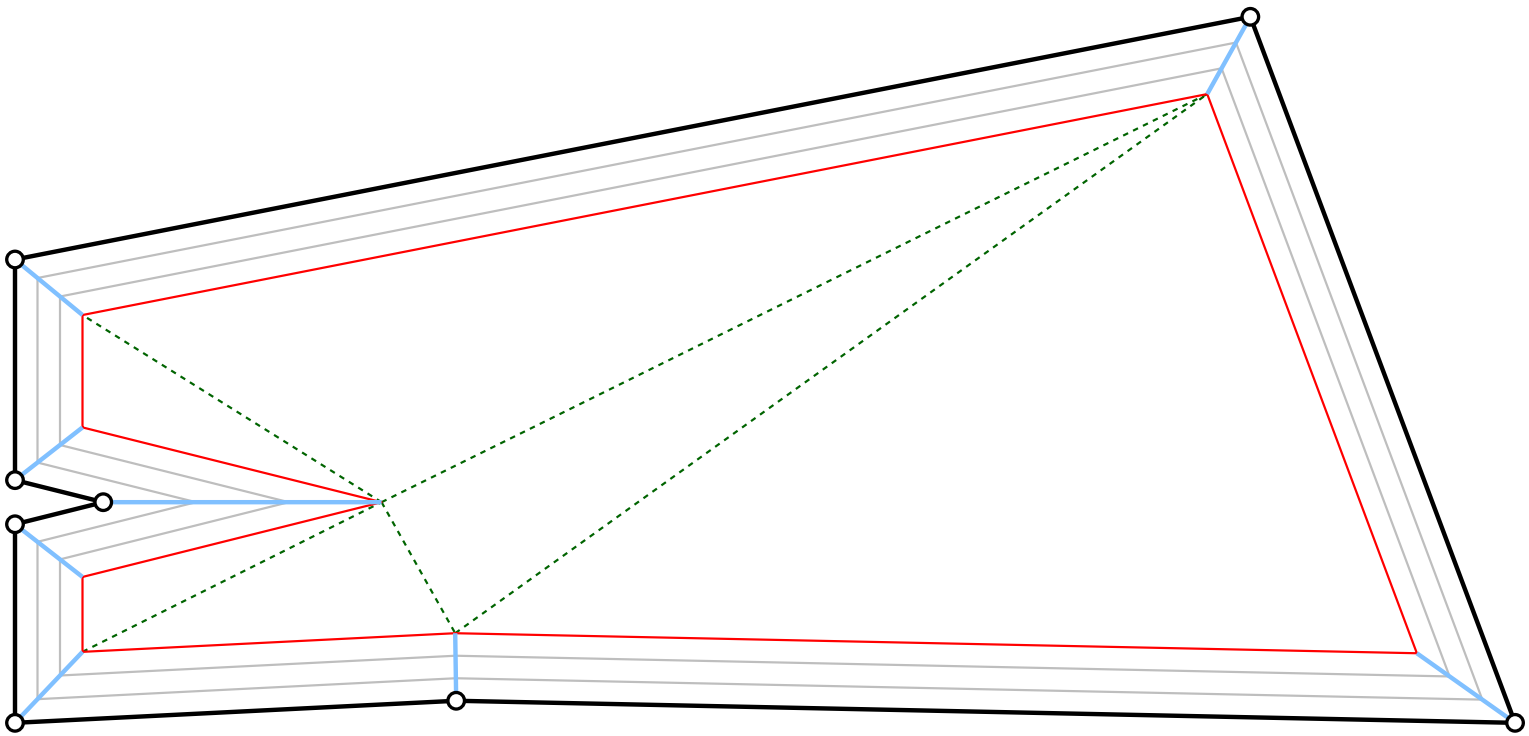
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.
- Instead they need special processing: *flip events*.



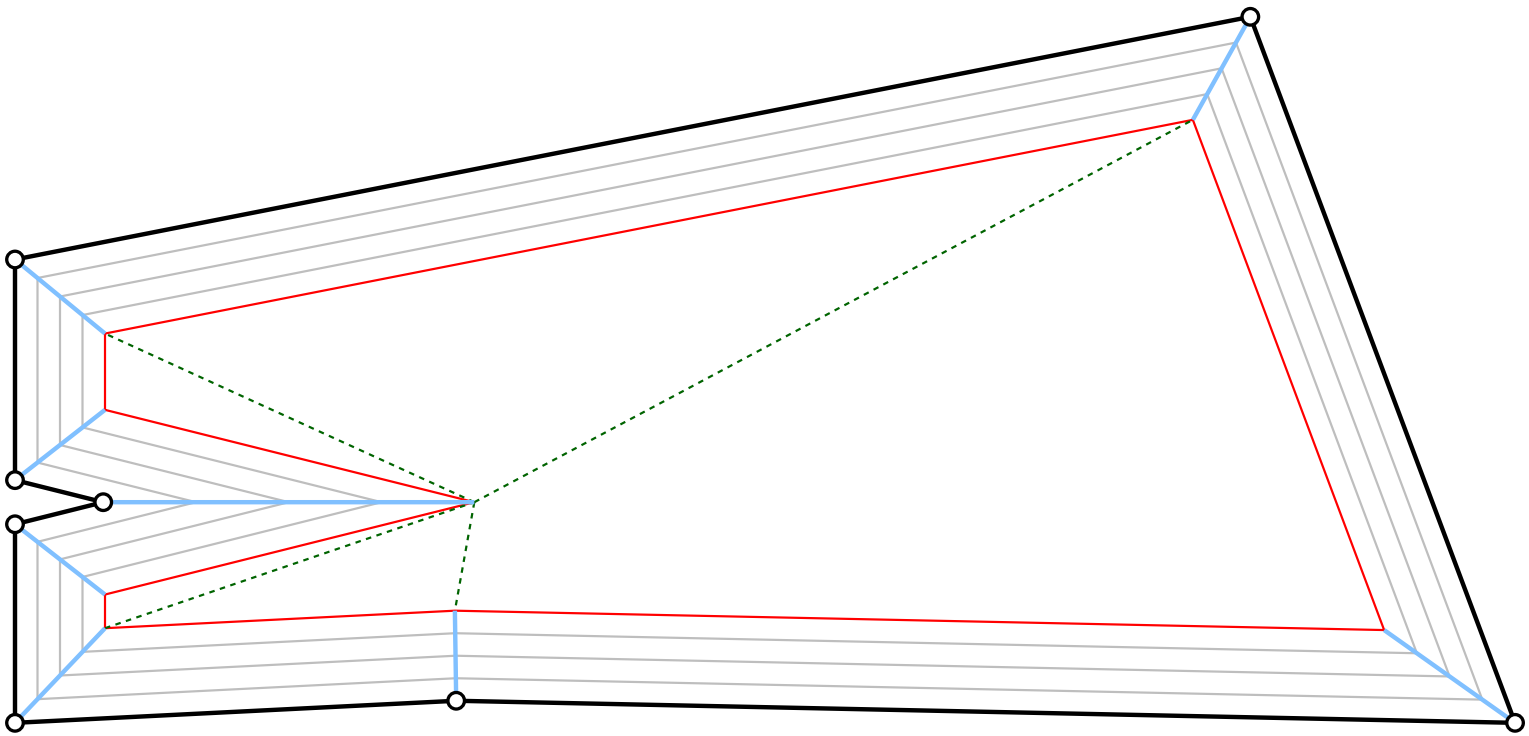
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.
- Instead they need special processing: *flip events*.



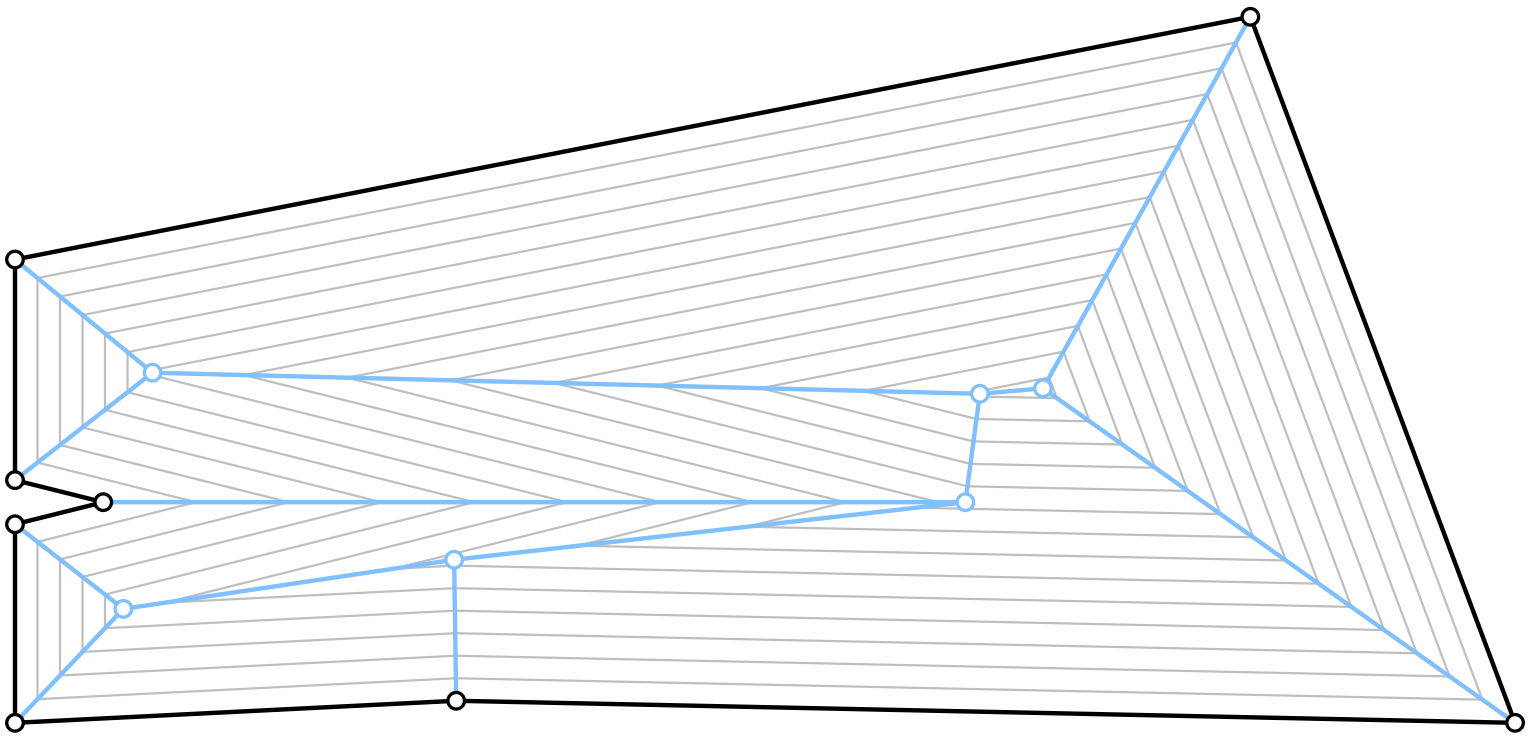
Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.
- Instead they need special processing: *flip events*.



Triangulation-based Algorithm

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored.
- Instead they need special processing: *flip events*.

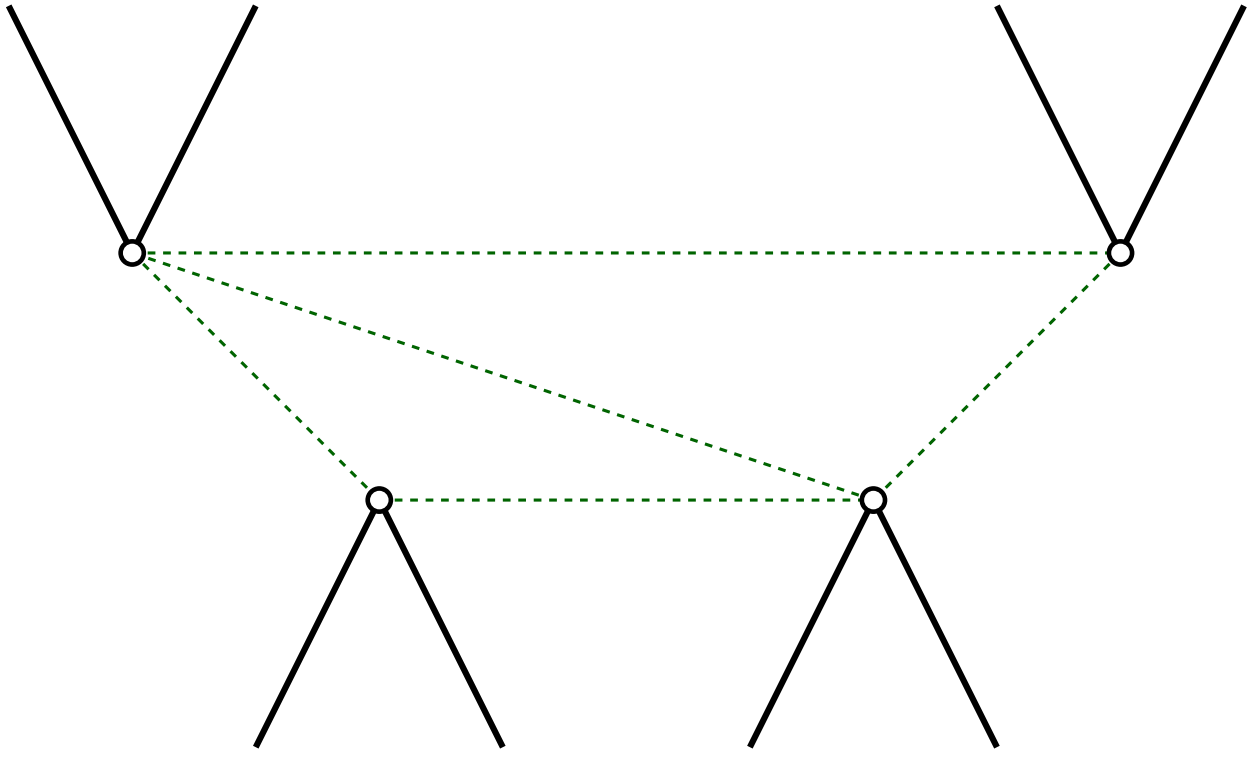


Contribution

- We have implemented this algorithm.
- We filled in gaps in the description of the algorithm.
- The algorithm does not always work when input is not in general position. We have identified and corrected these flaws.
- We have run extensive tests using this code.

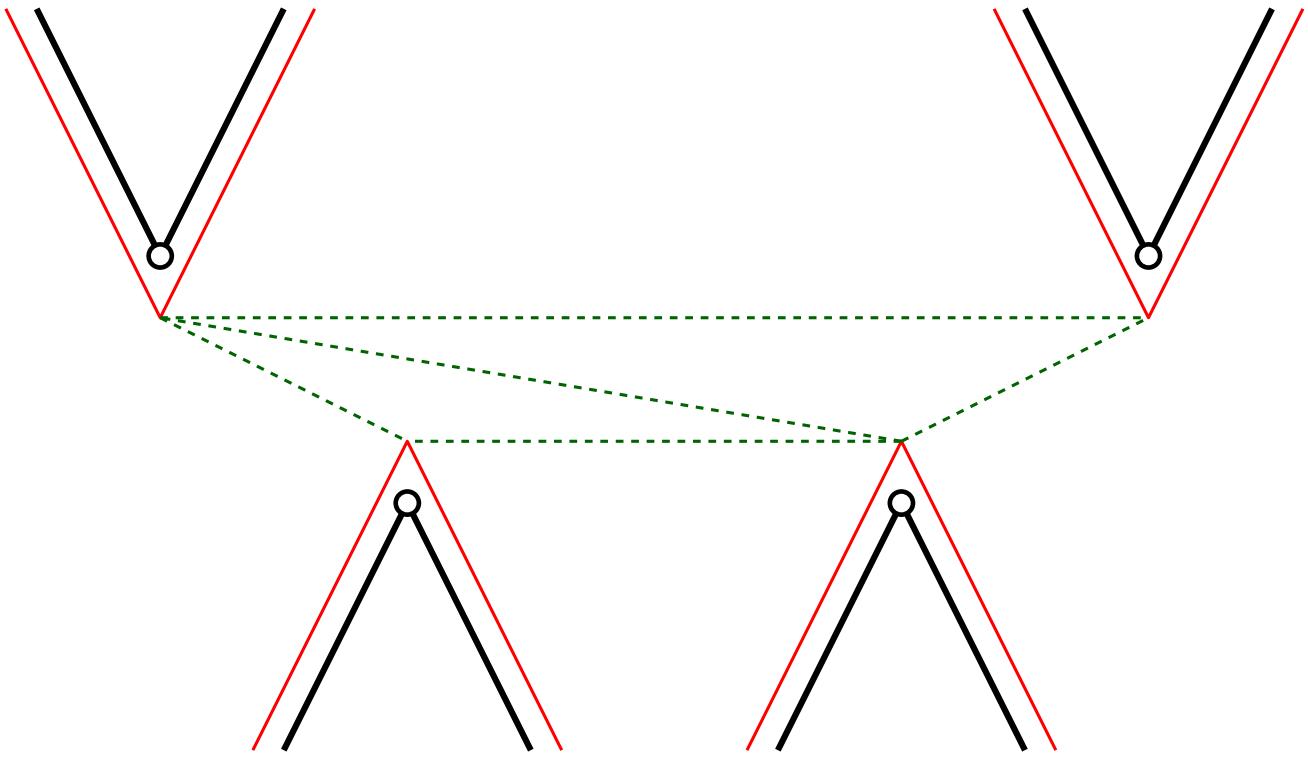
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



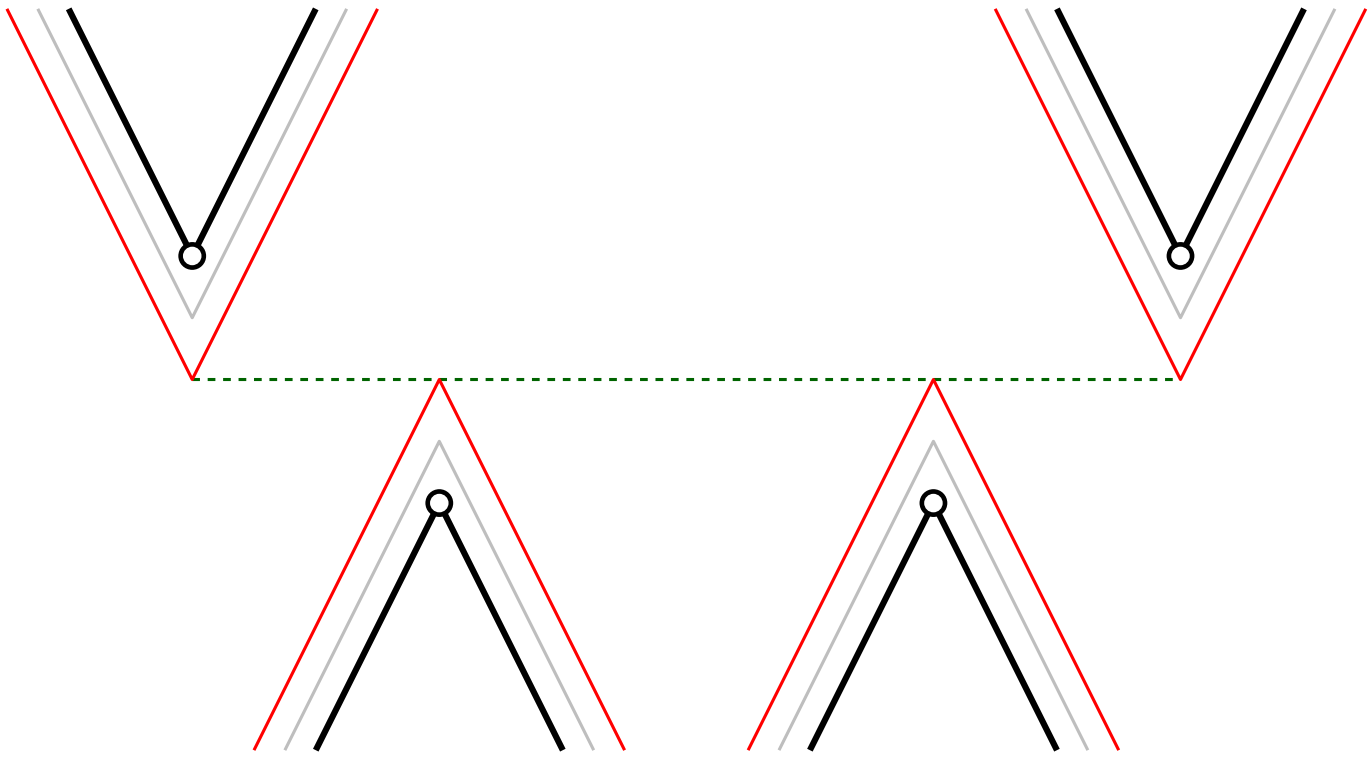
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



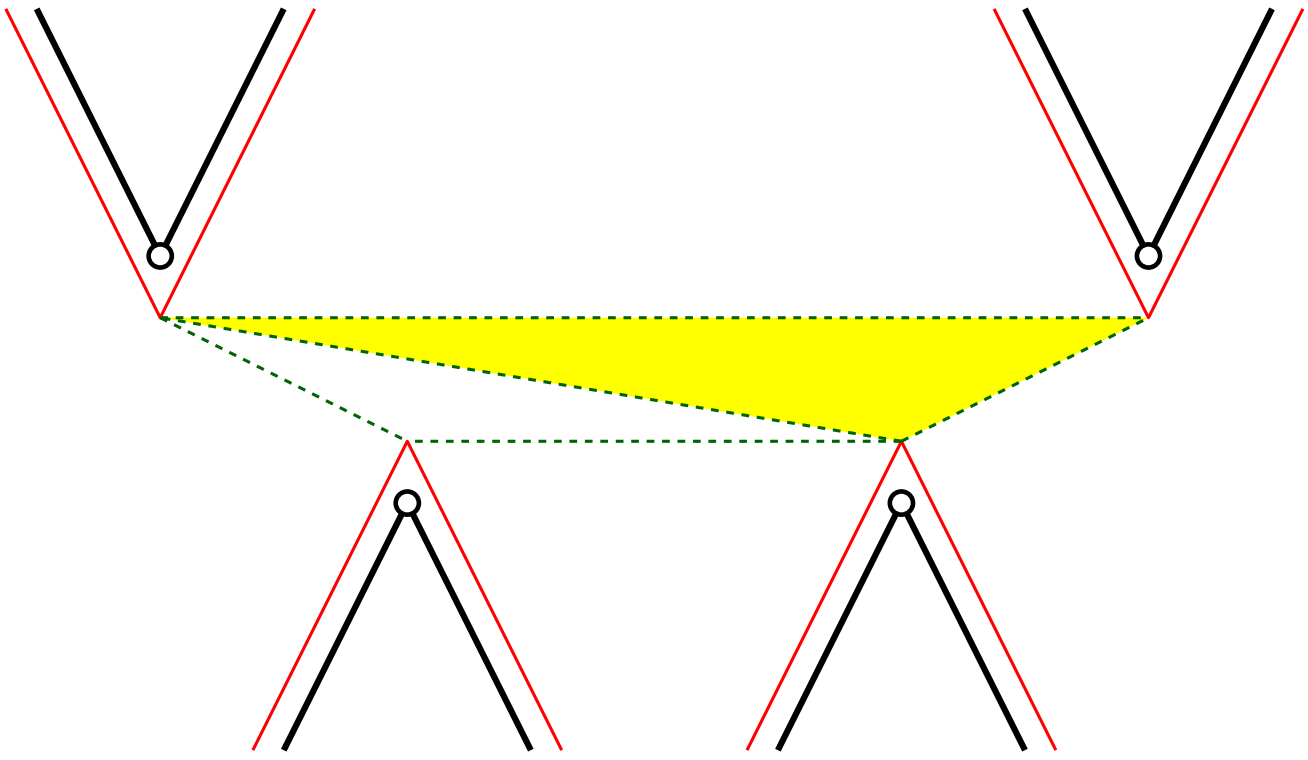
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



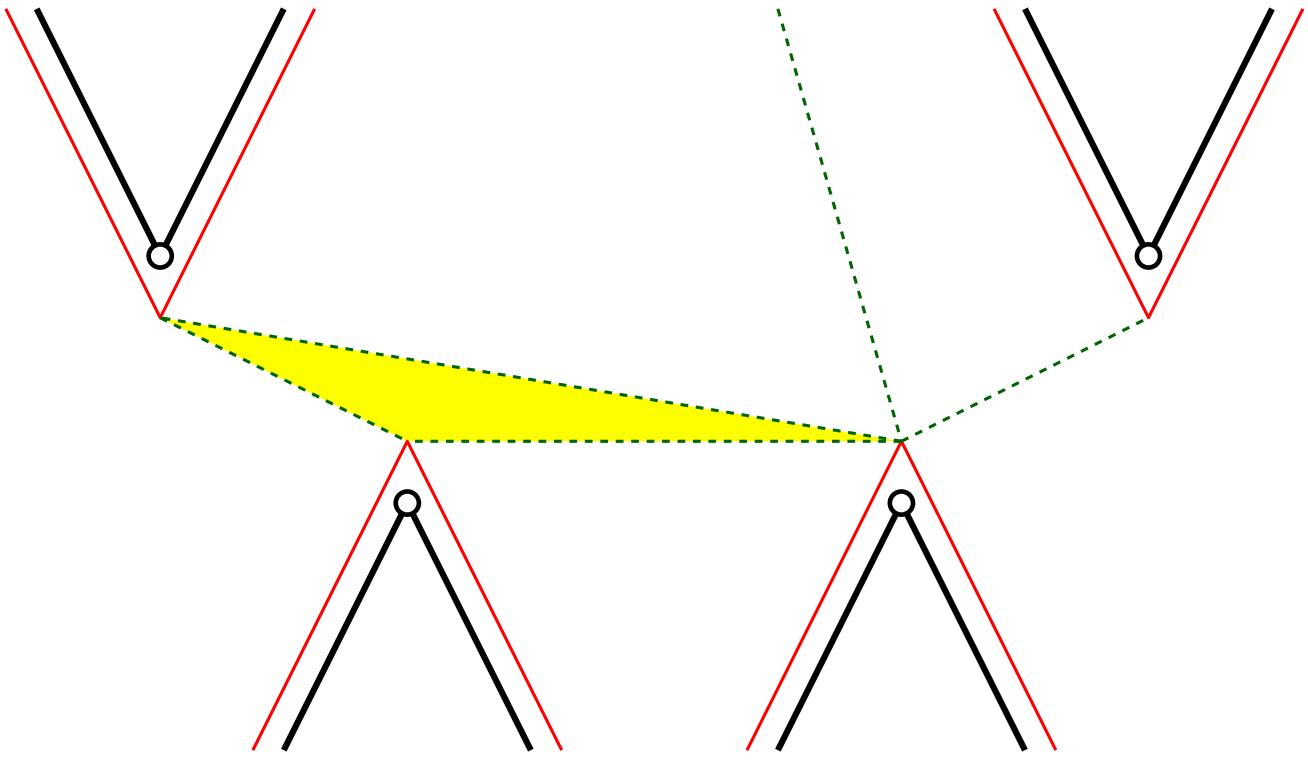
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



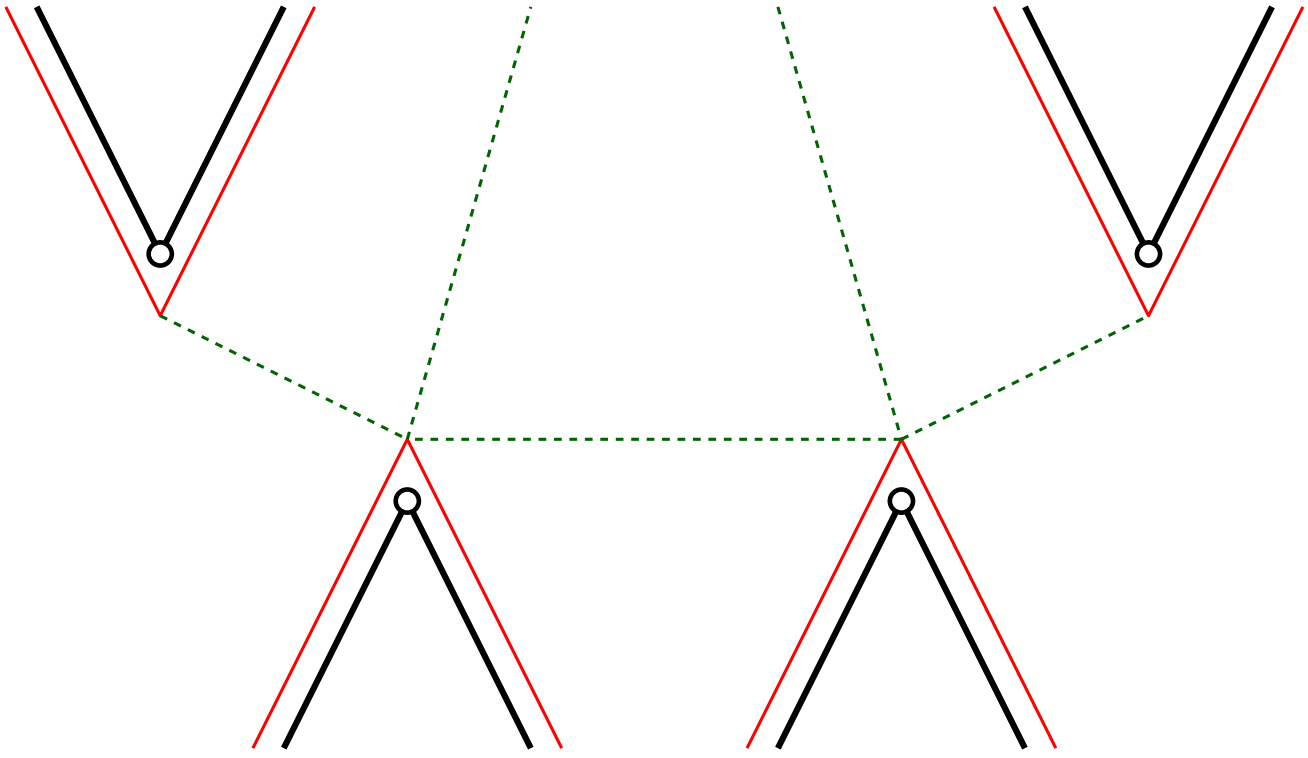
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



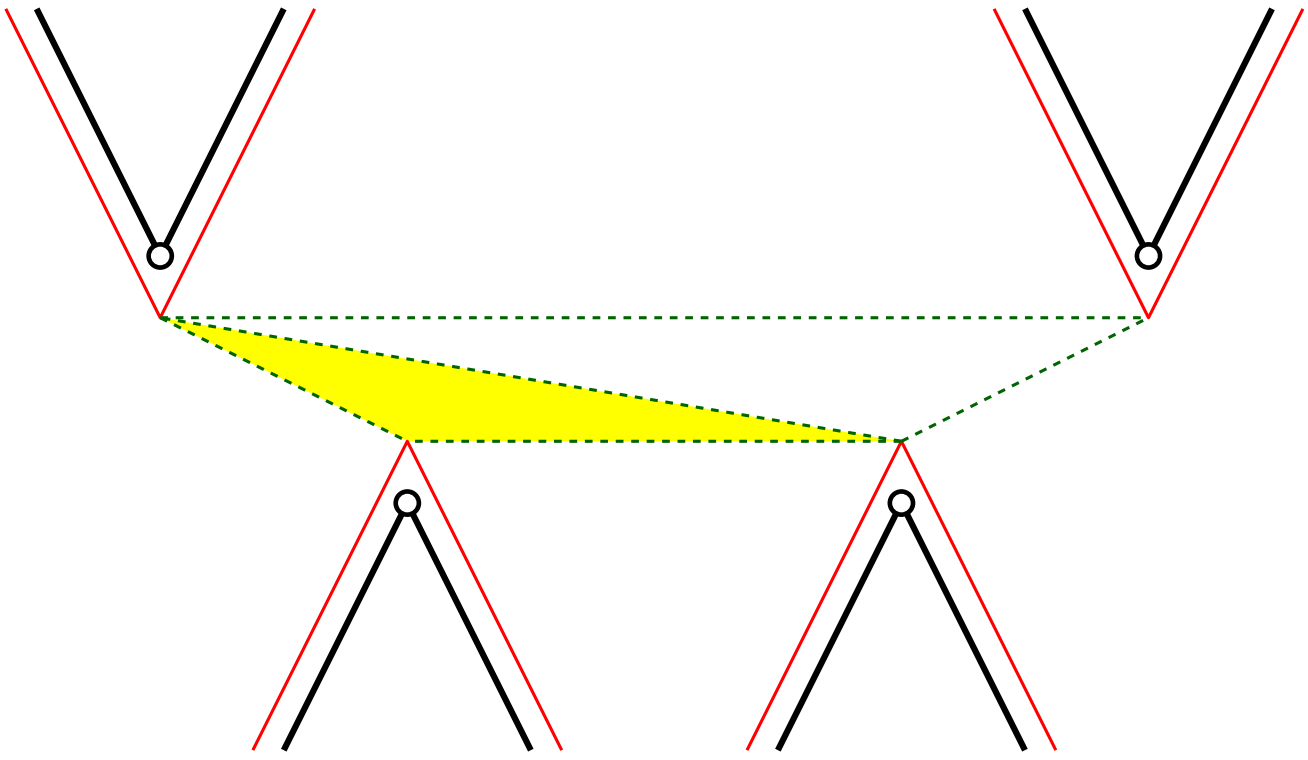
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



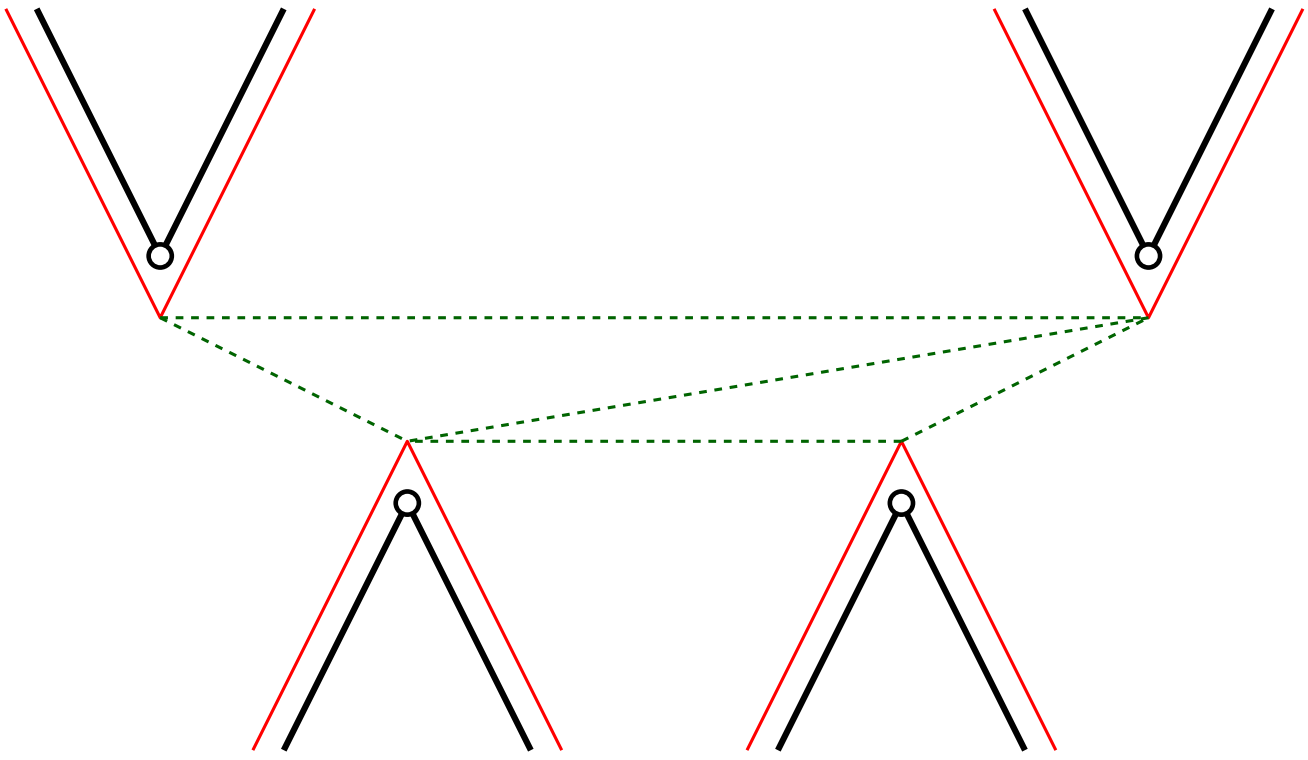
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



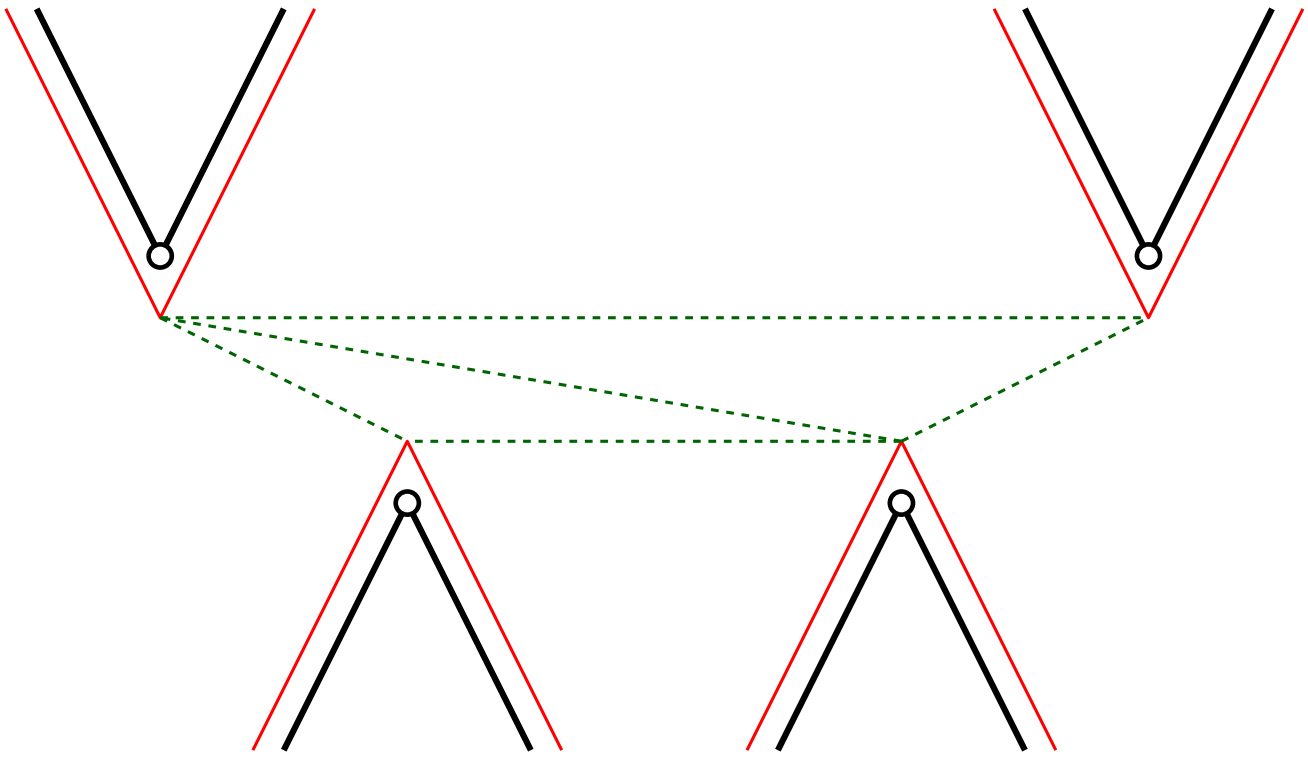
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



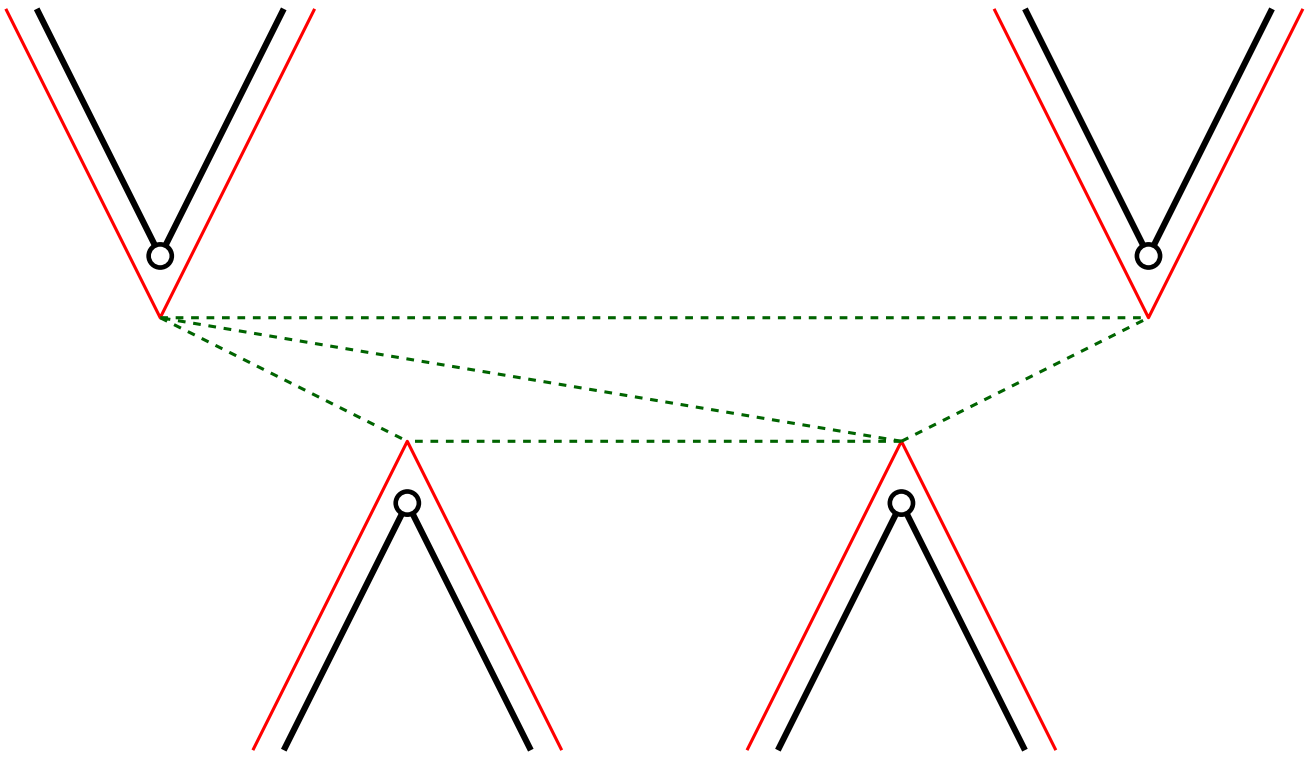
Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



Flip-Event Loops

- Without general position, this algorithm can end up in infinite loops.



- This is not a result of inexact floating point operations. The same can happen with exact arithmetic!

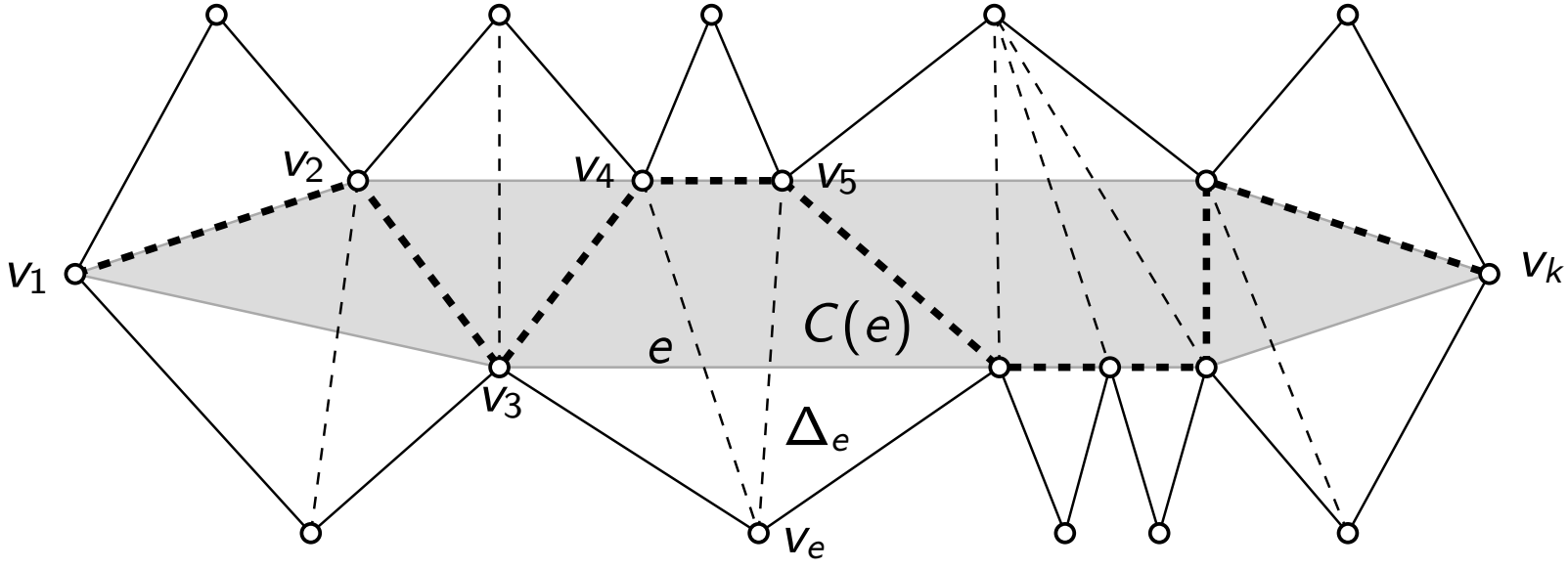
Detecting flip-event loops

- Keep a history of flip events $\langle e_1, e_2, \ldots \rangle$ where each $e_i = (t_i, \Delta_i)$.
- This history can be cleared when we encounter an edge or split event.
- If we encounter a flip event a second time, we may be in a flip-event loop.

Handling flip-event loops

Brief outline:

- Identify the polygon P which has collapsed to a straight line.
- Retriangulate P and its neighborhood.

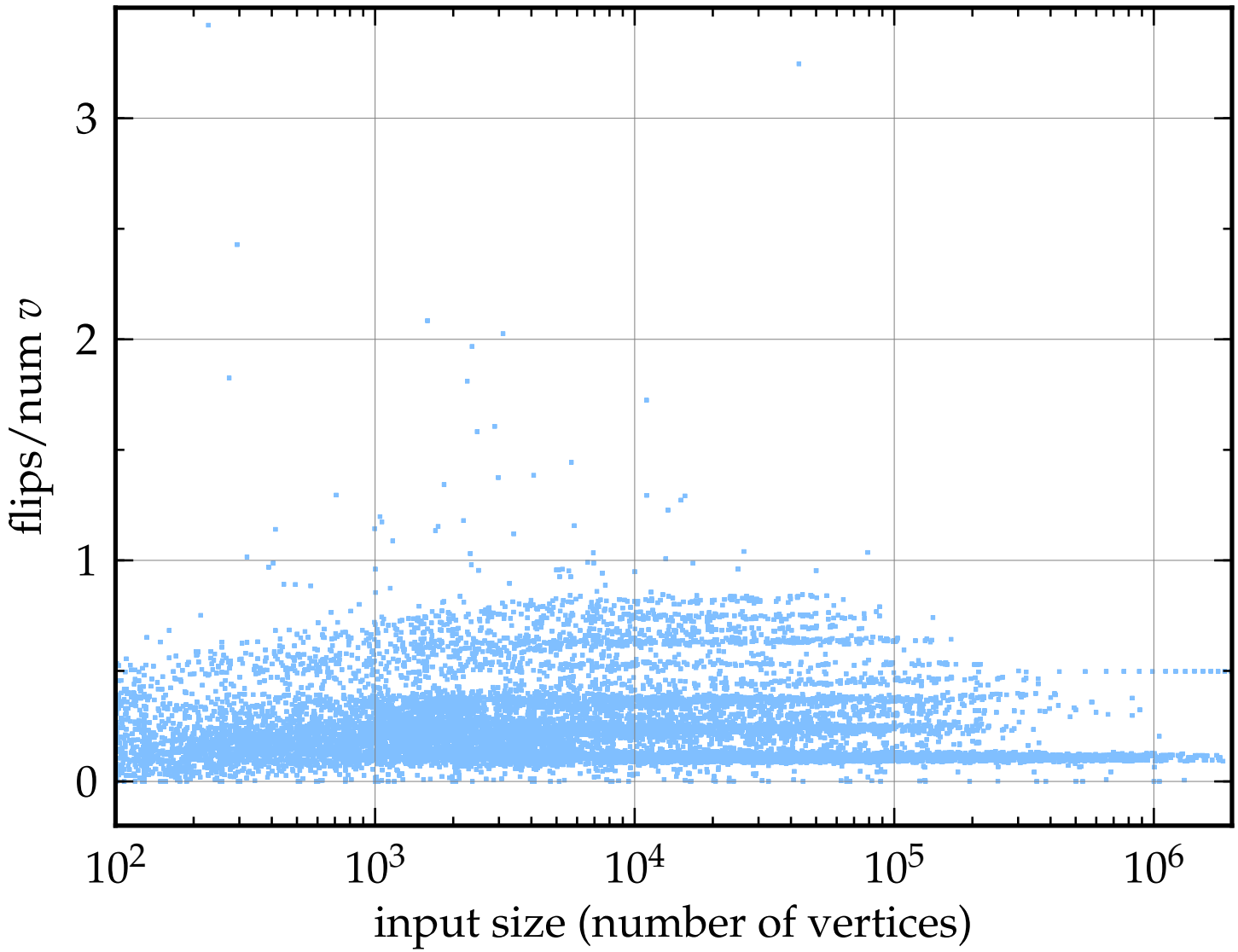


- This approach also is applicable to kinetic triangulations in other algorithms.

Number Of Flip Events

- $\mathcal{O}(n^3)$ is the best known upper bound on the number of flip events,
- No input is known that results in more than quadratically many flip events.
- It turns out that for *practical data* the number of flip events is very linear.

Number of Flip Events, II

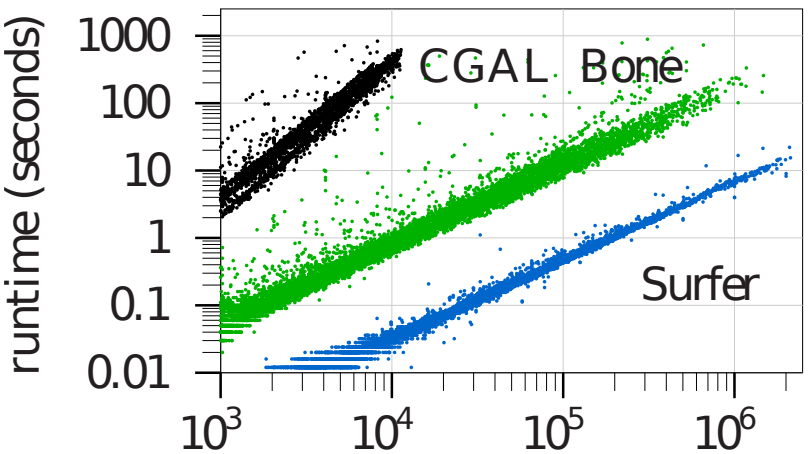


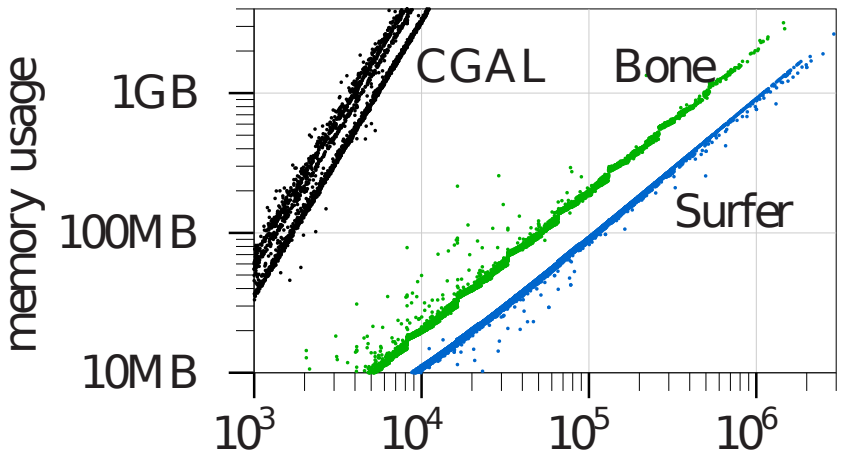
Performance observations

	theoretical worst case		practical	
	runtime	space	runtime	space
E&E[1]	$\mathcal{O}(n^{17/11+\epsilon})$	$\mathcal{O}(n^{17/11+\epsilon})$	N/A	
CGAL[2]	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^2)$
Bone[3]	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$
Surfer[4]	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$

[1] Eppstein and Erickson, 1999
[2] F. Cacciola, 2004
[3] Huber and Held, 2010
[4] `this`, based on Aichholzer and Aurenhammer, 1998

RUNTIME TESTS





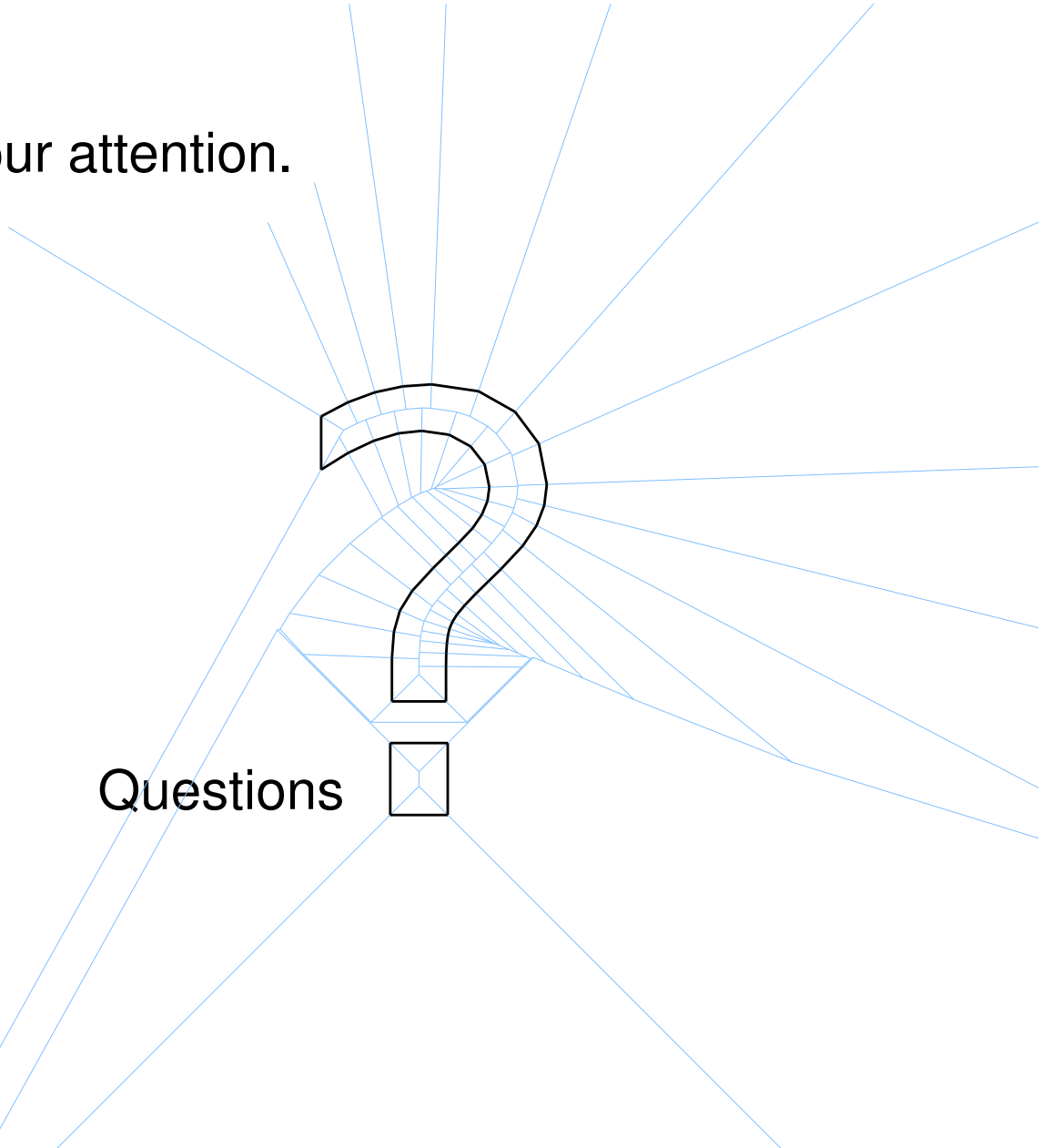
Runtime and memory usage behavior of CGAL, Bone, and Surfer for inputs of different sizes.
Bone and Surfer use their IEEE 754 double precision backend.

Summary

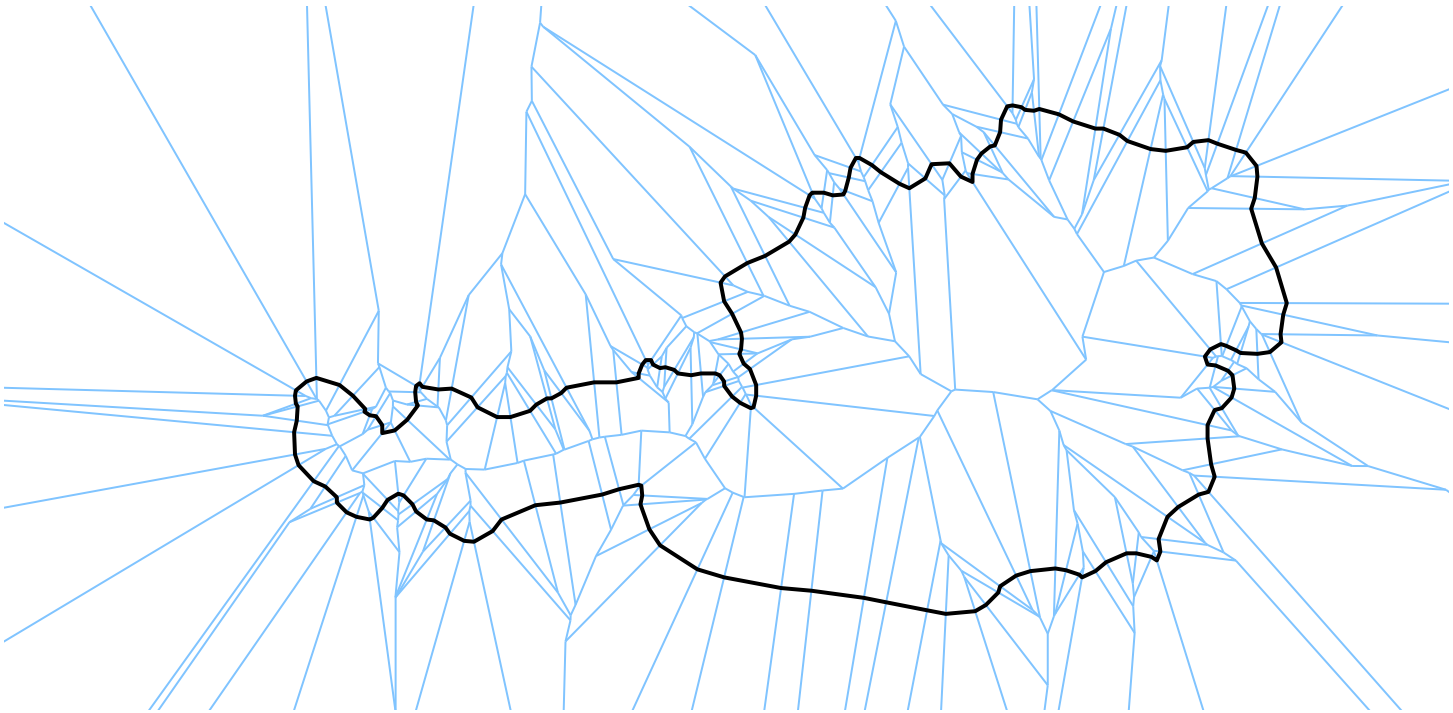
- We have implemented Aichholzer and Aurenhammer's algorithm from 1998, filling in details in the algorithm description.
- We fixed real problems that arise in the absence of general position.
- Our approach to handling flip events has wider applications.
- The implementation runs in $\mathcal{O}(n \log n)$ time for *real-world data*. The number of flip events is linear in practice.
- It is industrial-strength, having been tested on tens of thousands of inputs.
- It is the fastest straight skeleton construction code to date, handling millions of vertices in mere seconds.

Questions

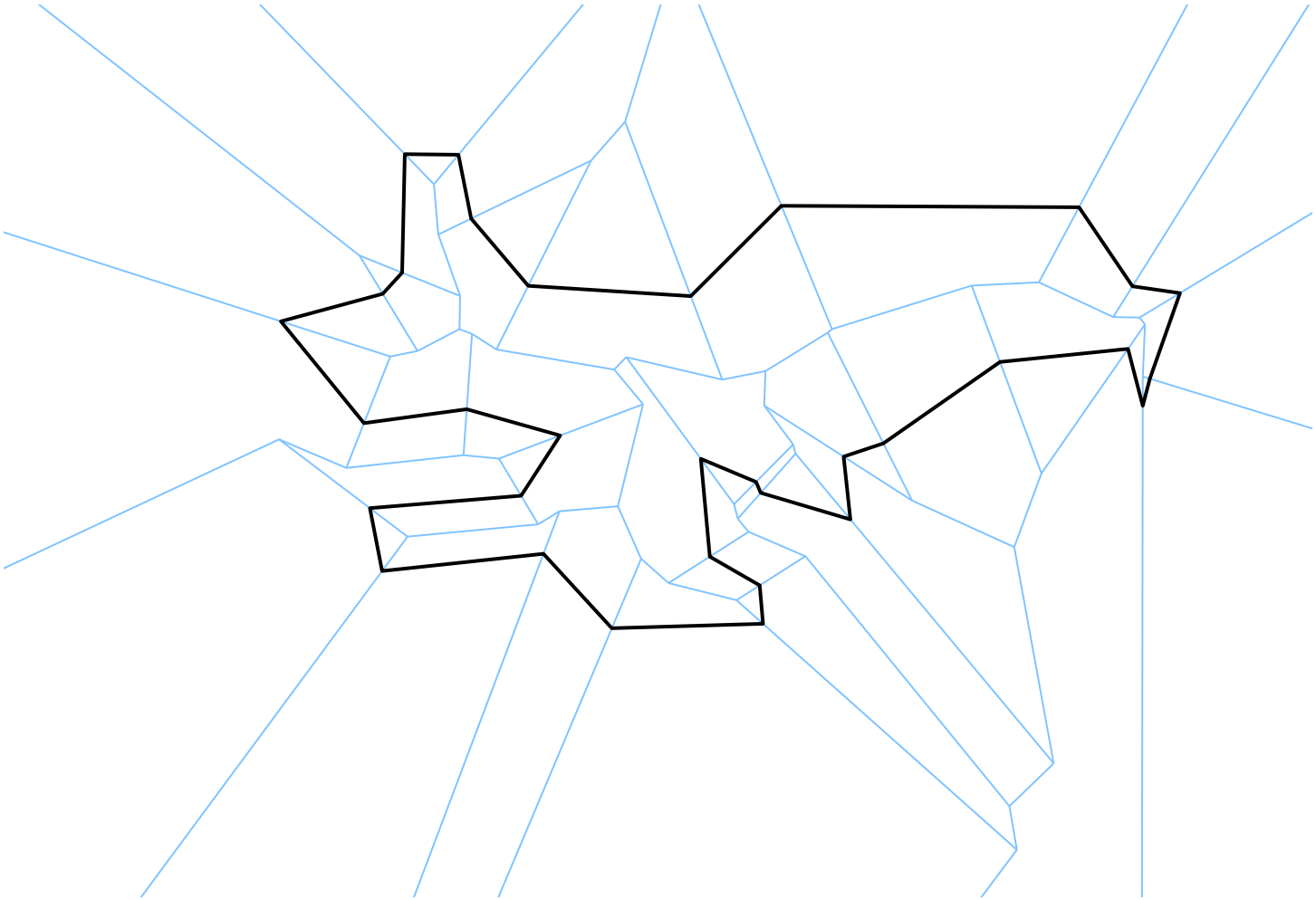
Thank you for your attention.



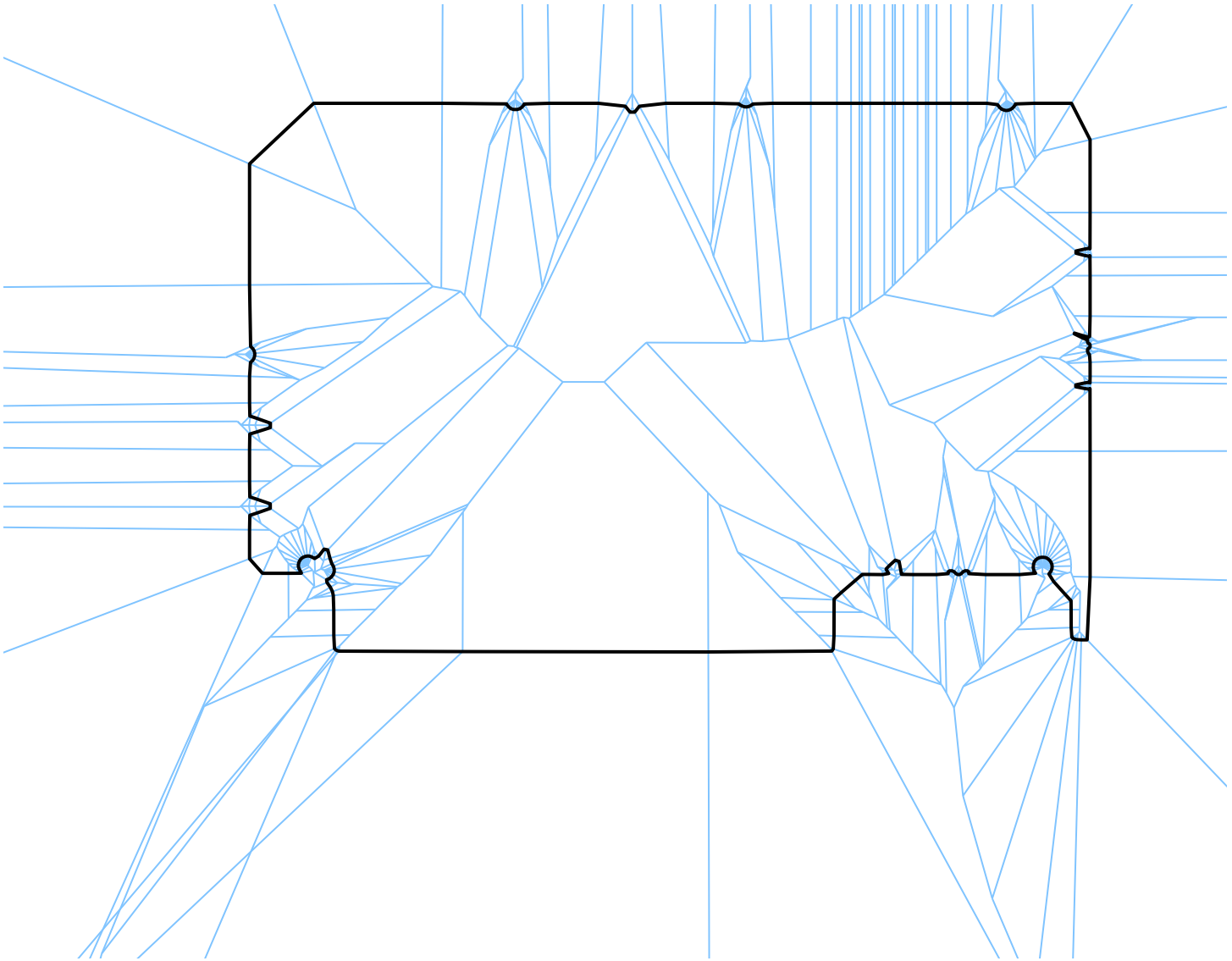
Gallery: Borders of Austria



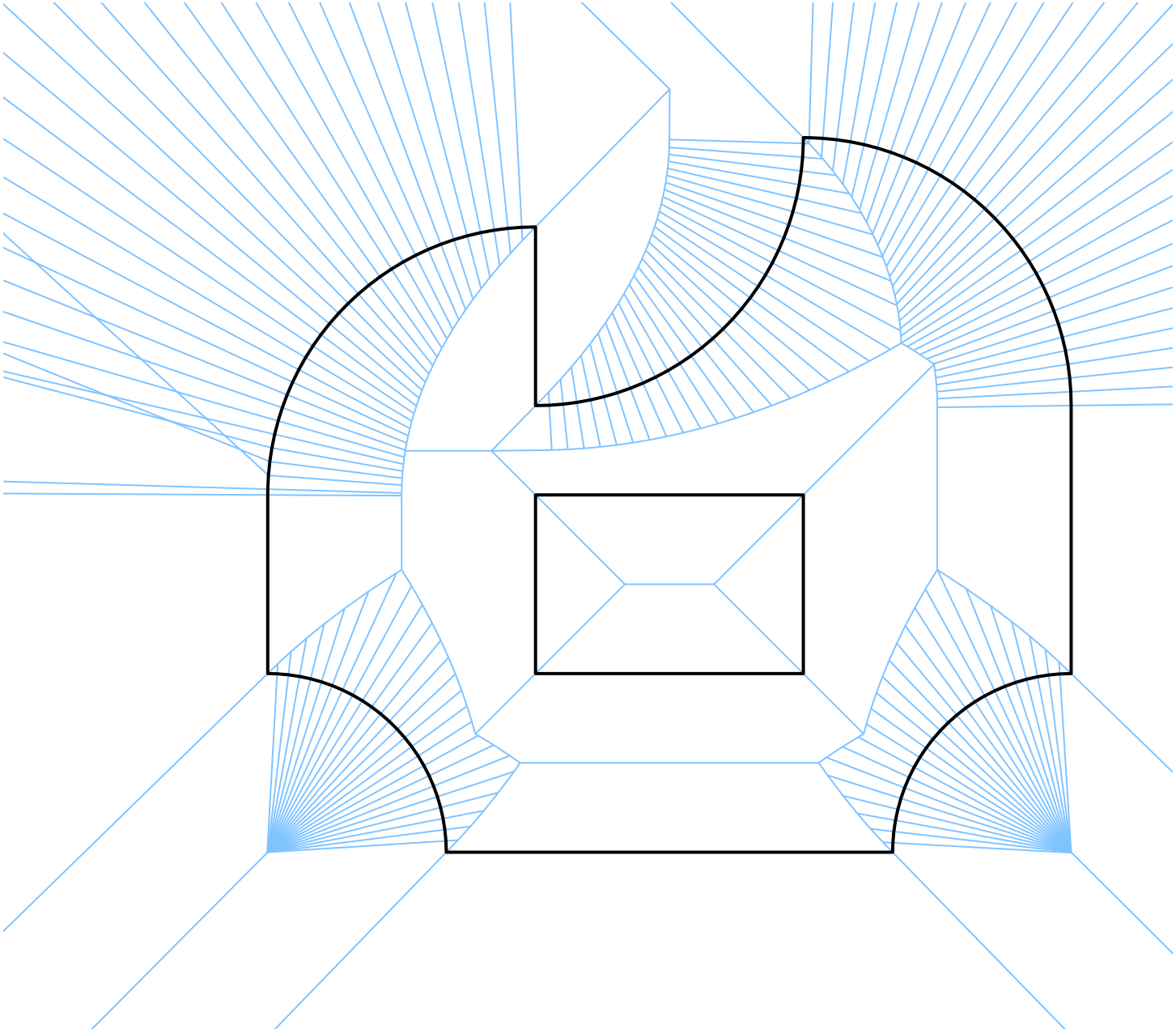
Gallery: Random Polygon



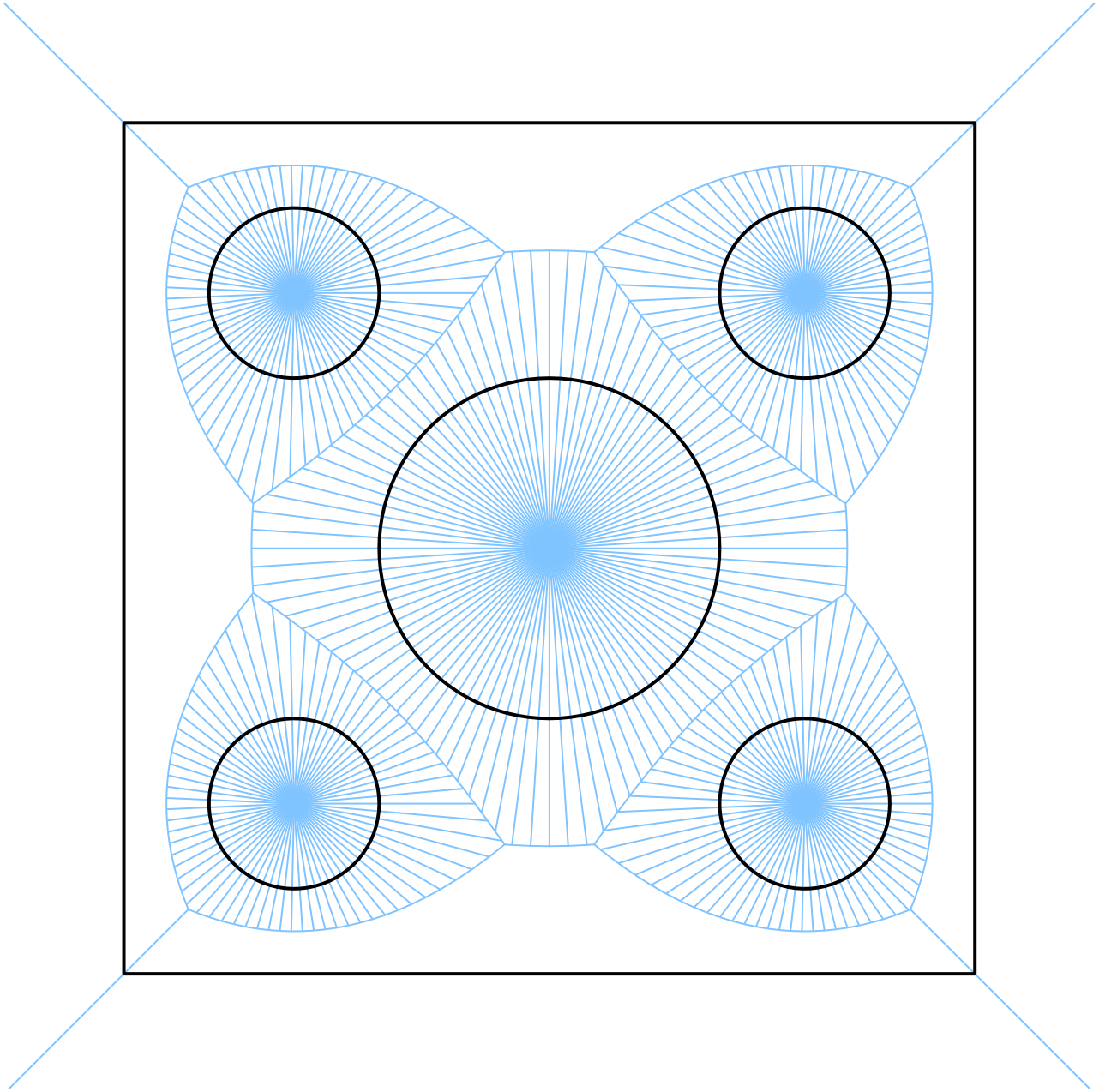
GALLERY: PCB



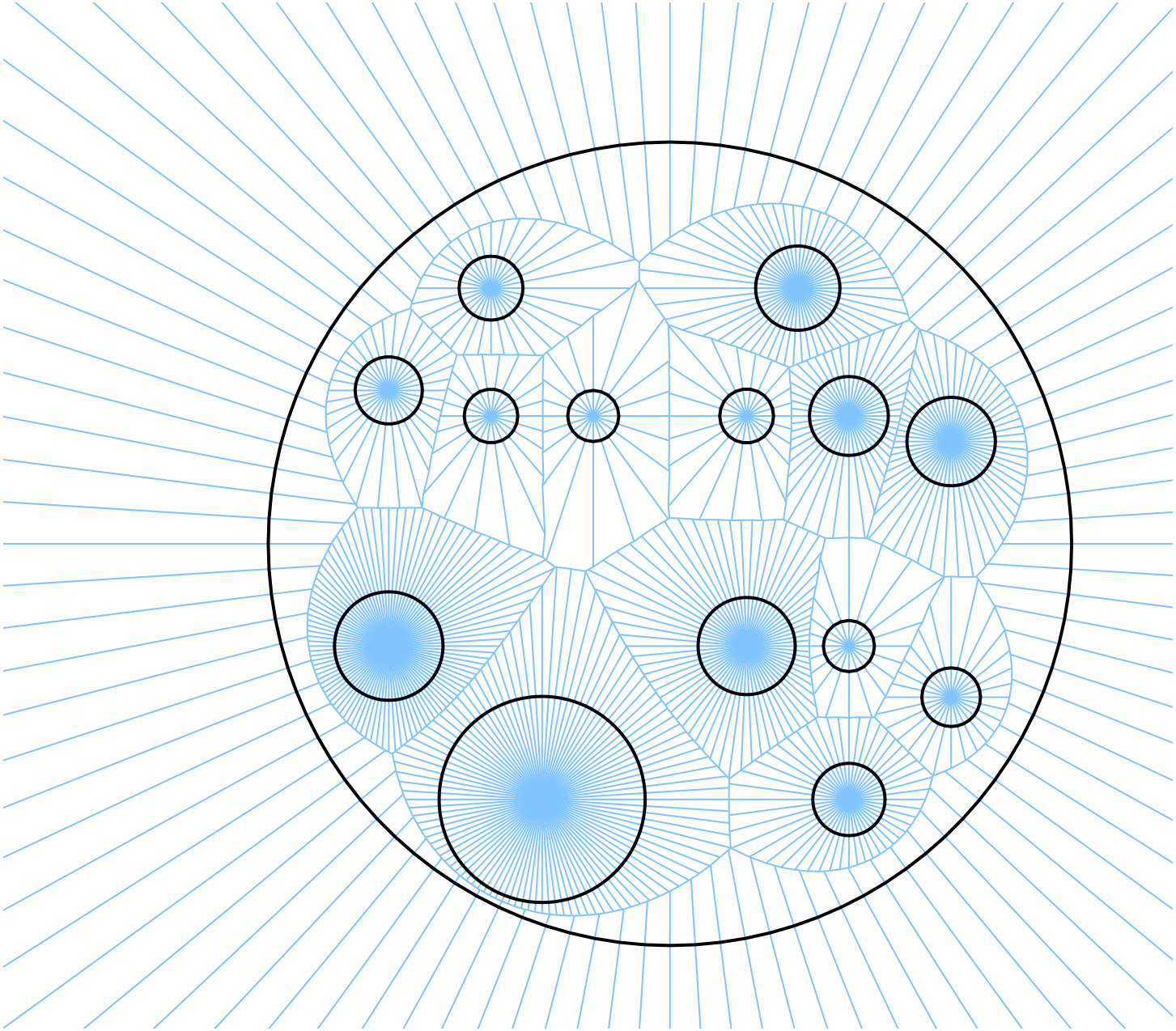
Gallery: Polygon with hole



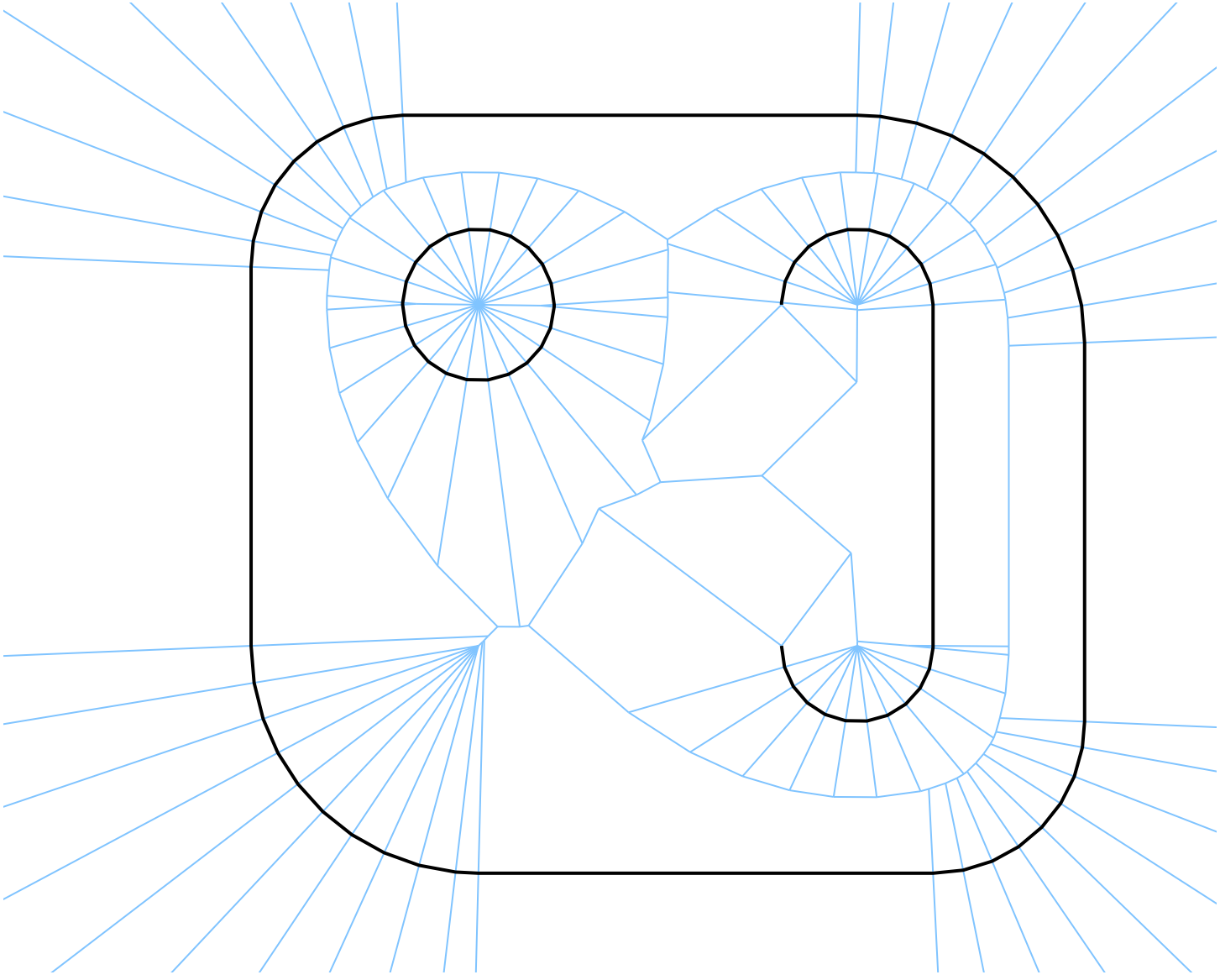
Gallery: Circular Holes



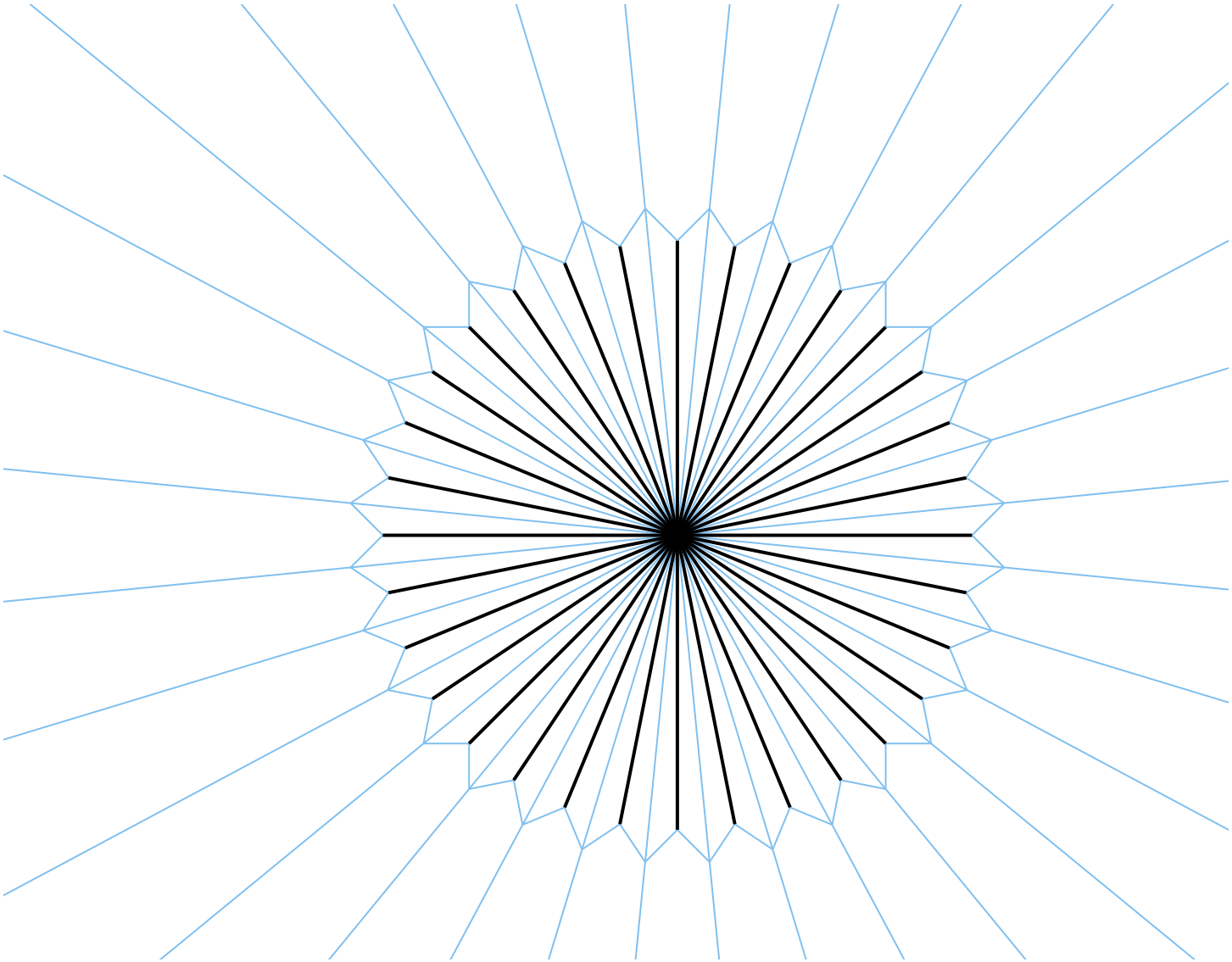
Gallery: More Holes



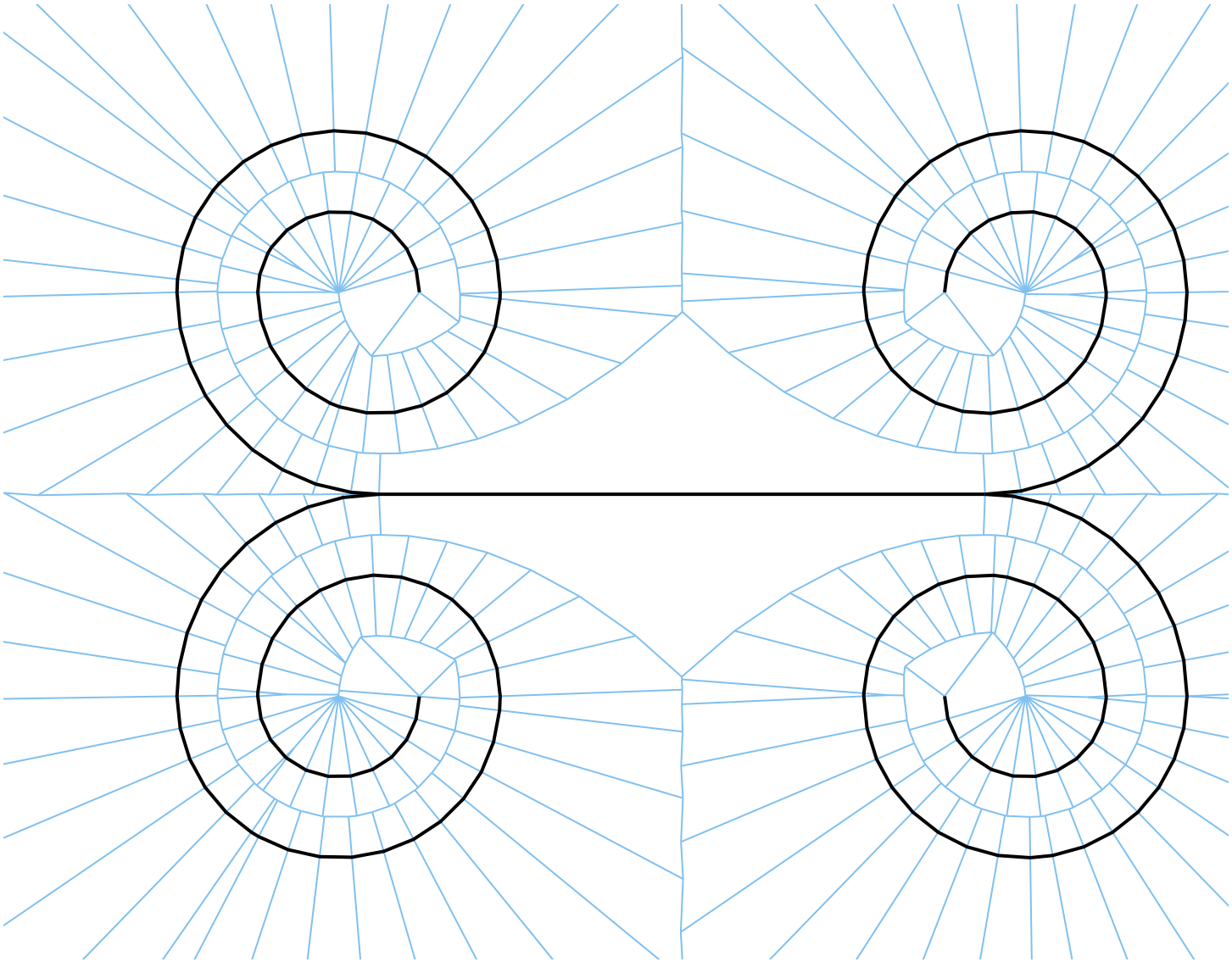
Gallery: Almost Polygon



Gallery: Star



Gallery: Spirals



Applications: GIS

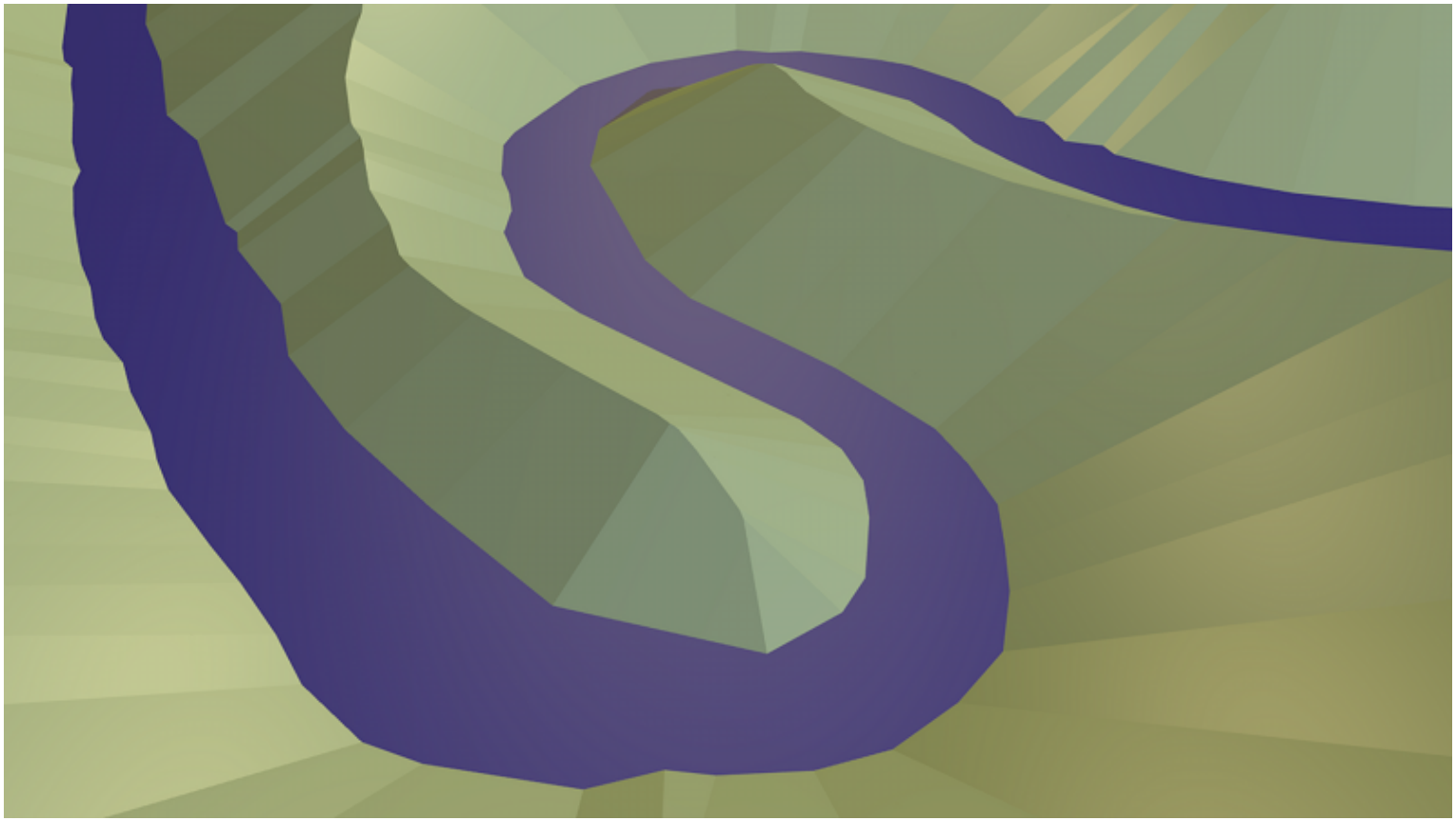
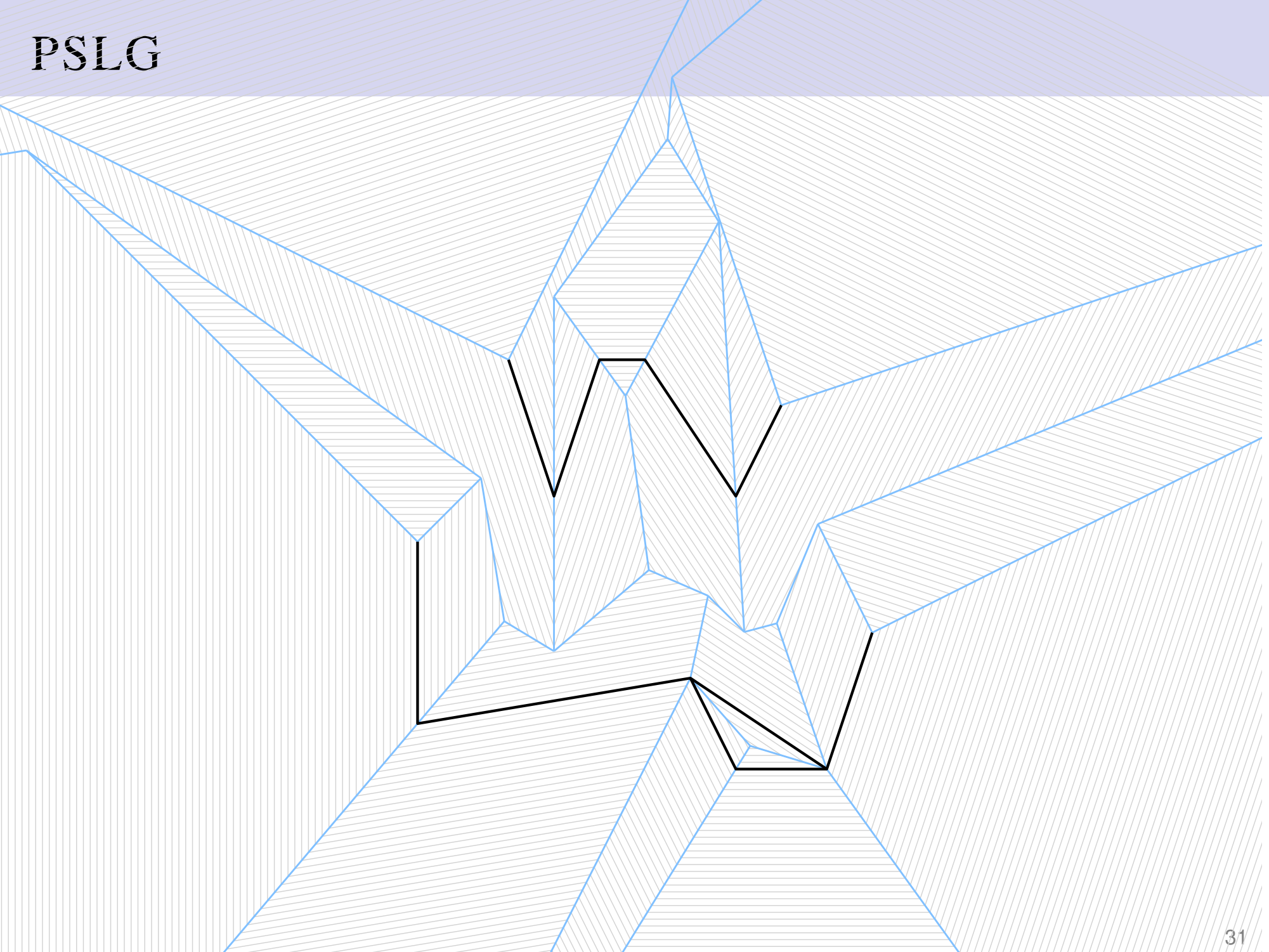
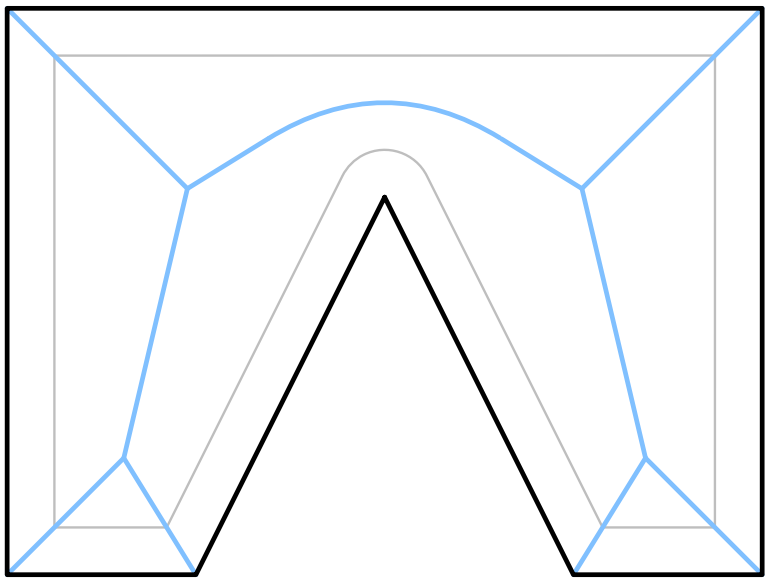


image credit: Stefan Huber

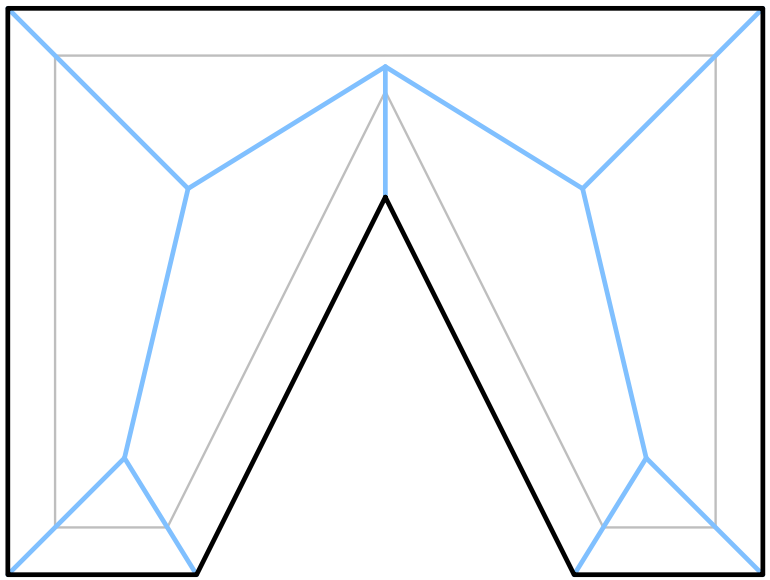
PSLG



Medial Axis vs. SK

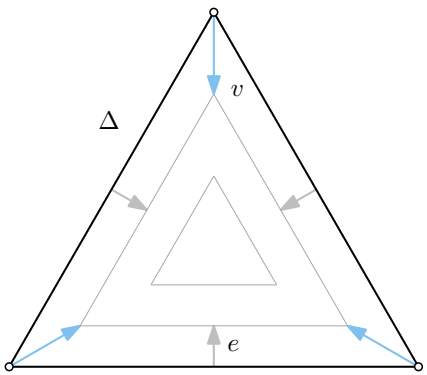


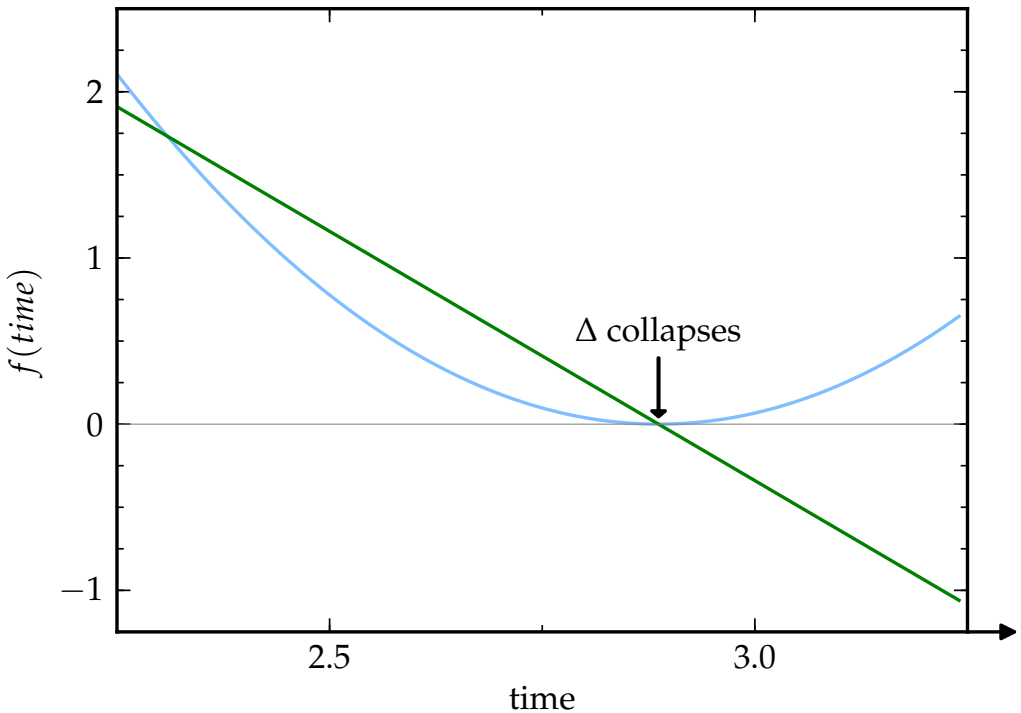
VD-based MA



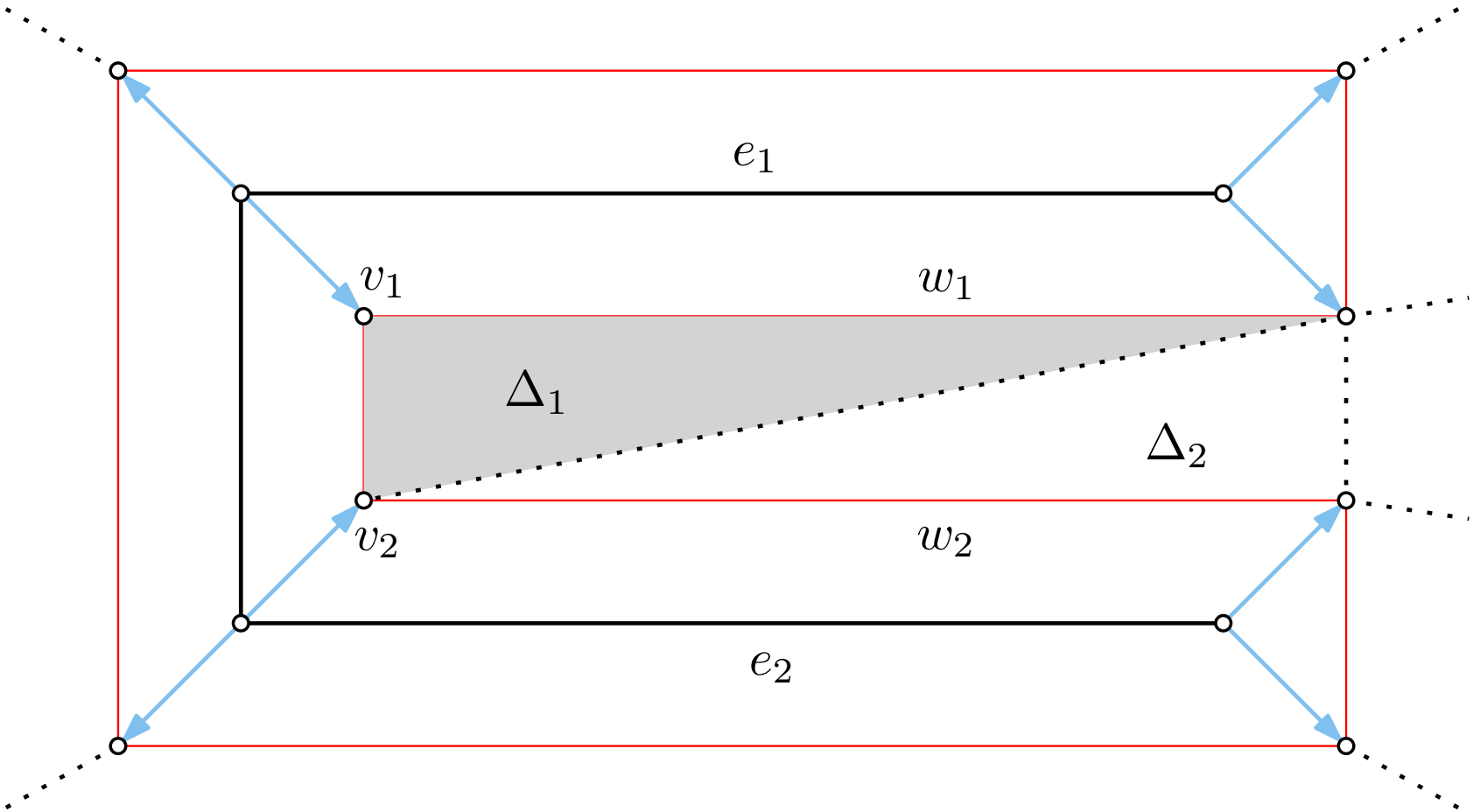
SK

Alternate Computation



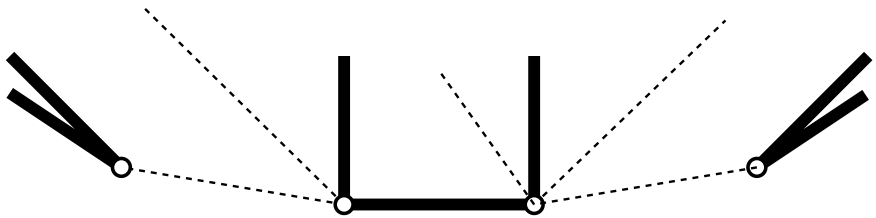


Infinitely fast vertices



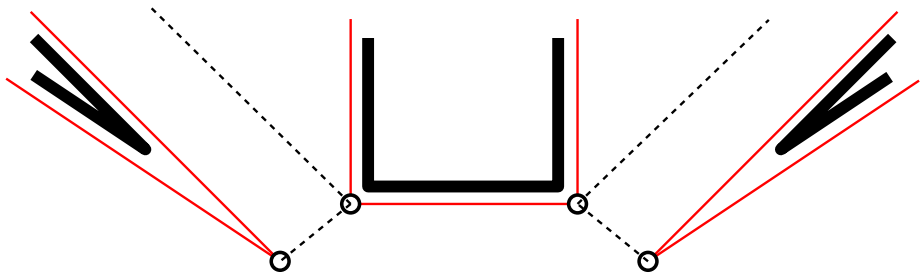
Triangulating

- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



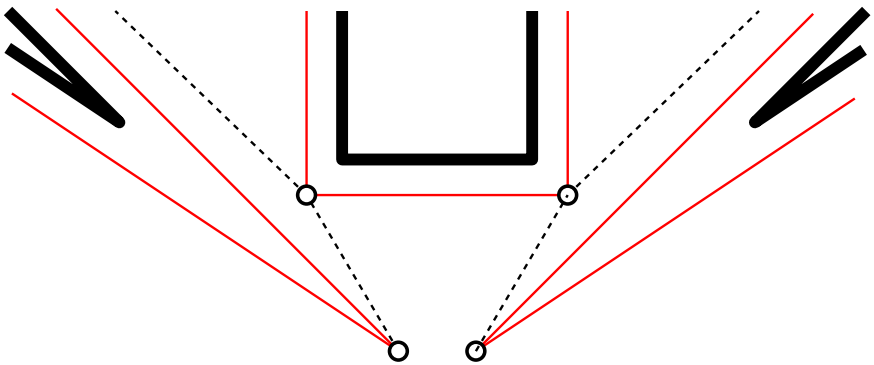
Triangulating

- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



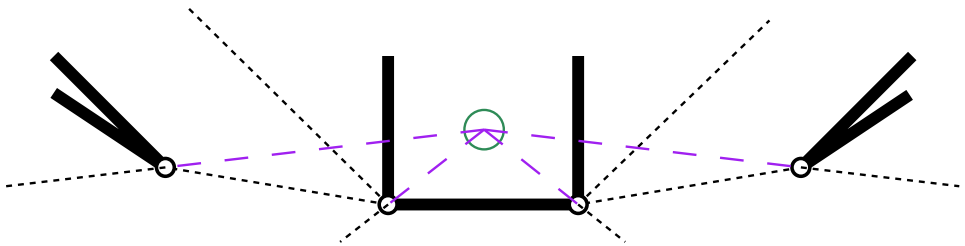
Triangulating

- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



Triangulating

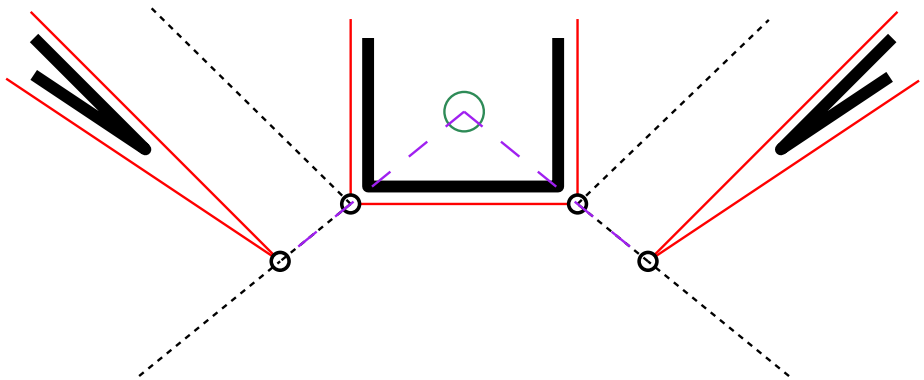
- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



- We need to update the triangulation at some point before this happens, but how?

Triangulating

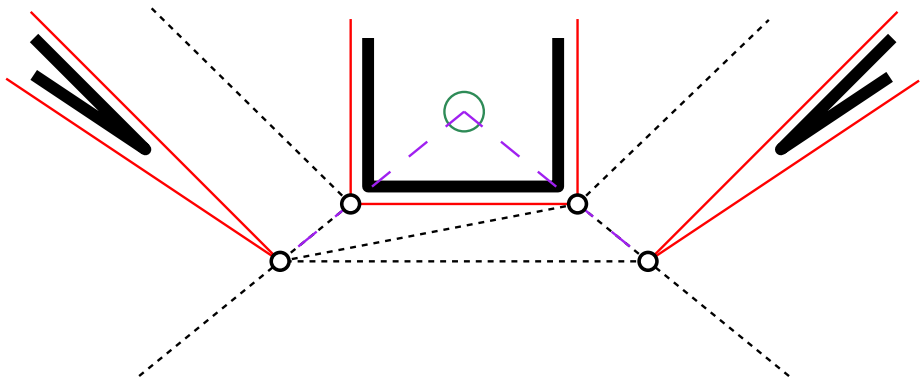
- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



- We need to update the triangulation at some point before this happens, but how?

Triangulating

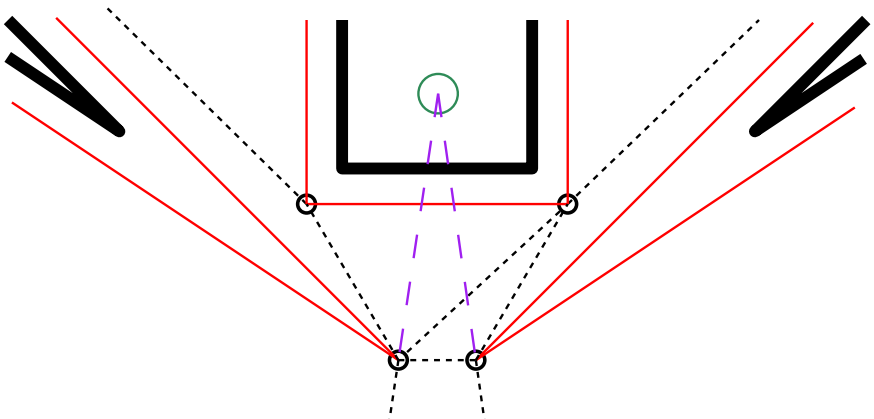
- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.



- We need to update the triangulation at some point before this happens, but how?

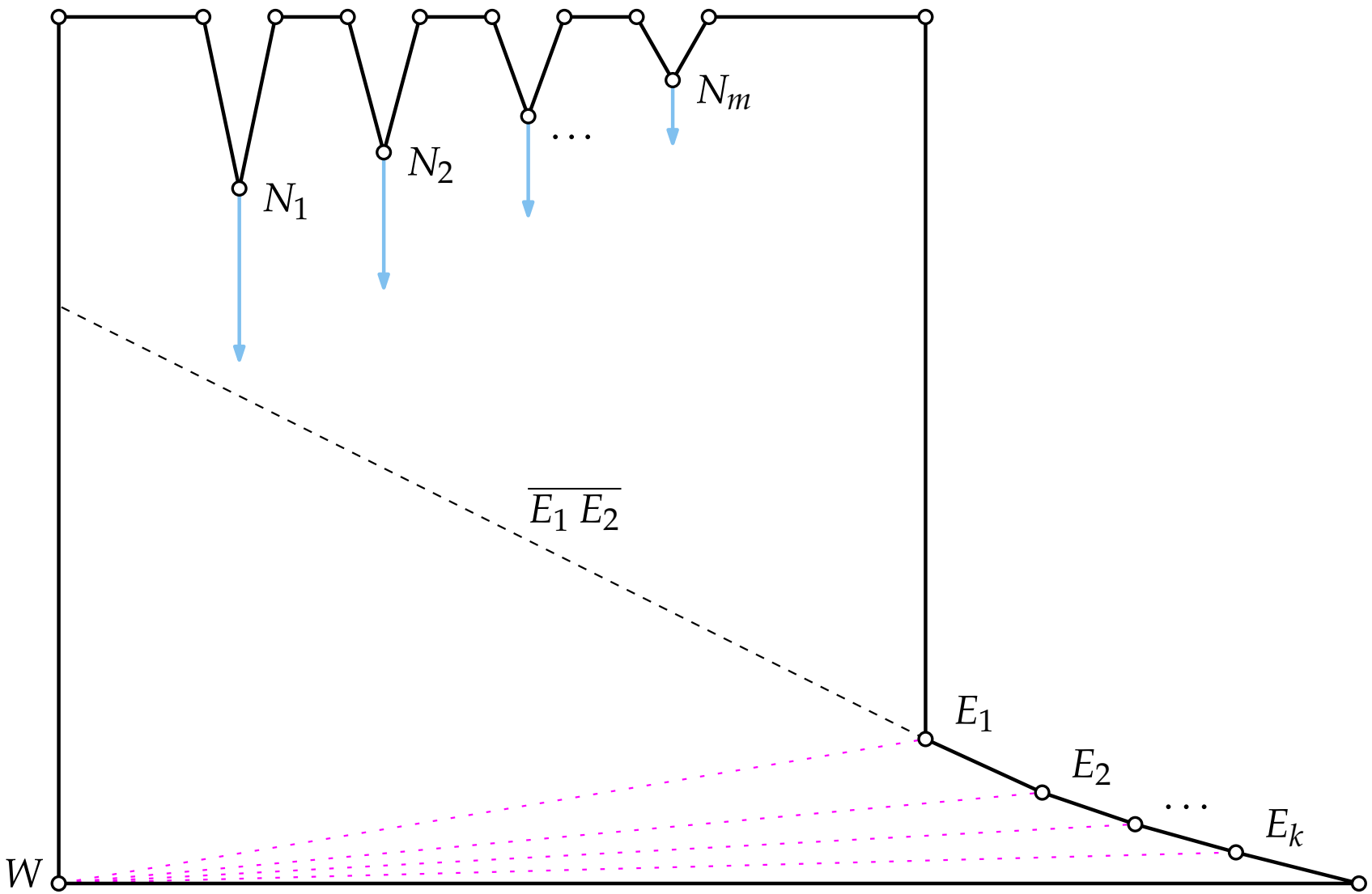
Triangulating

- Triangulate the convex hull.
- Unfortunately the convex hull changes with time, and it matters.

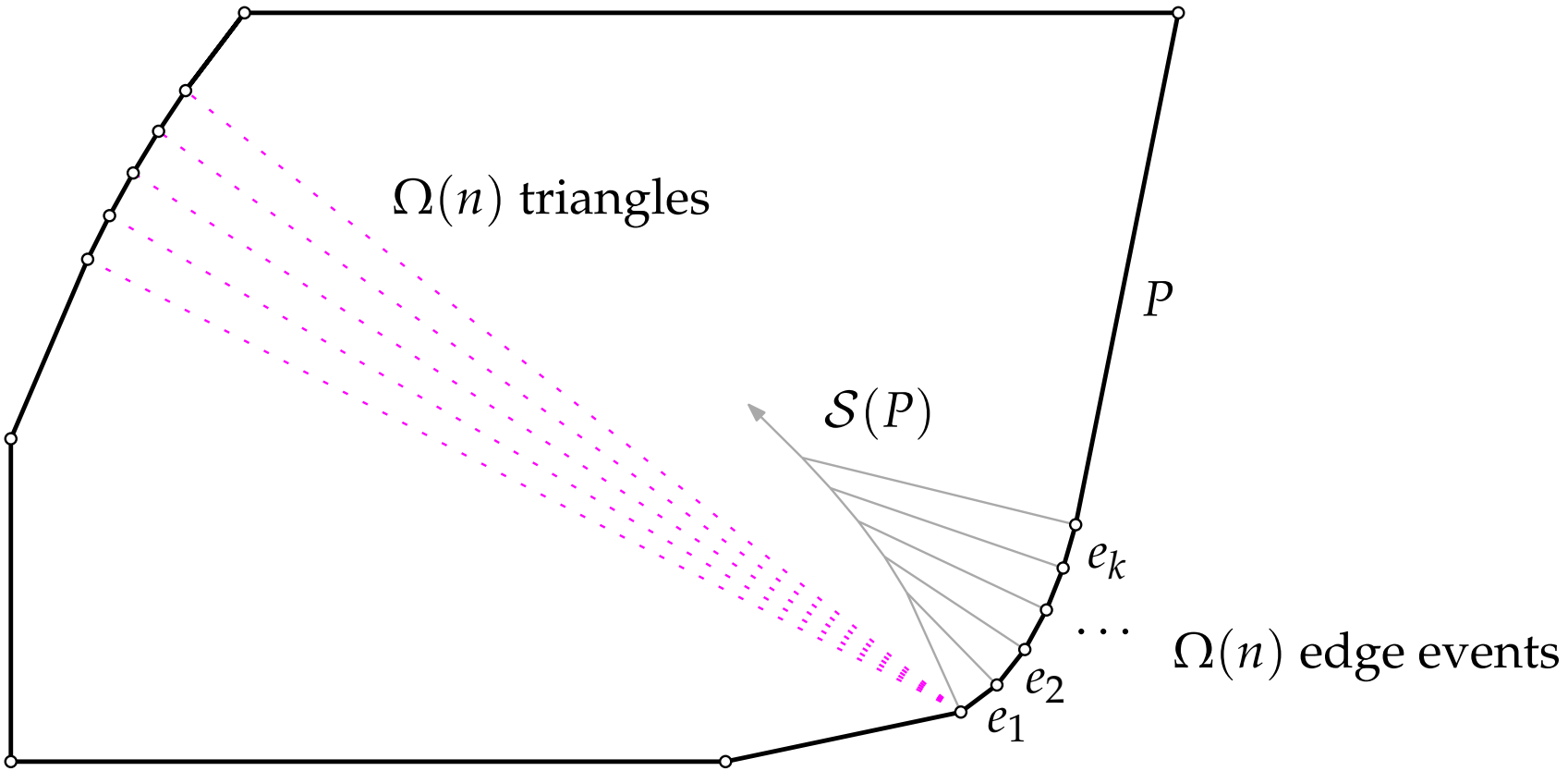


- We need to update the triangulation at some point before this happens, but how?

Ω FOR FLIP EVENTS



Ω FOR NON-FLIP EVENTS

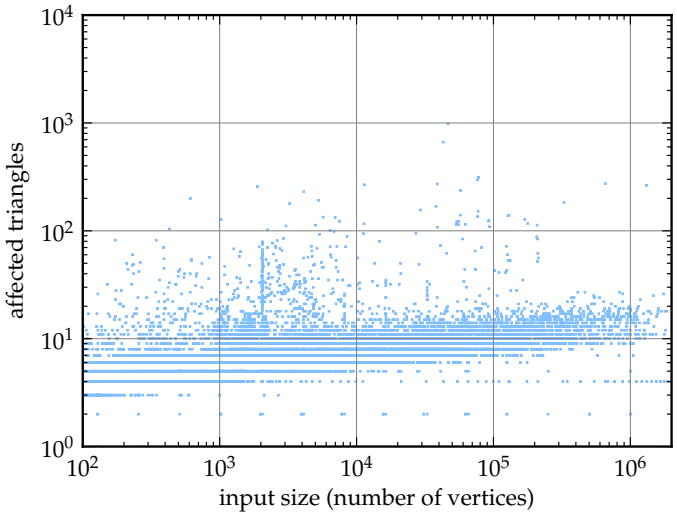


Solution for Flip-Event Loops with EGC

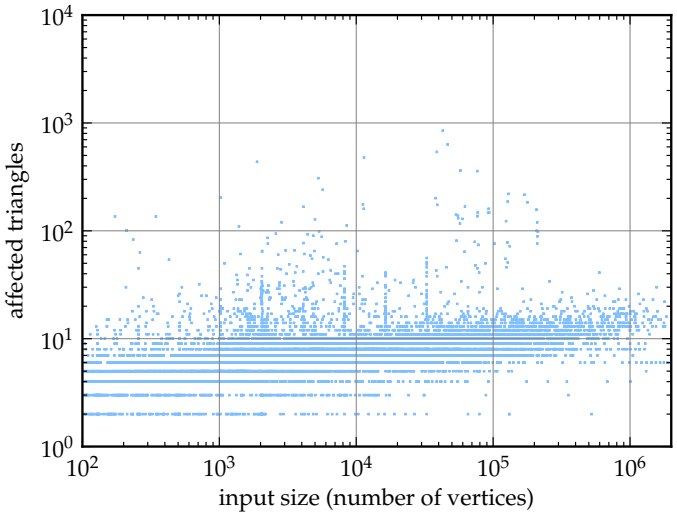
Pick, in order:

- non-flip event $\rightarrow$ reduces triangles
- longest edge to flip $\rightarrow$ reduces longest edge (count or length)

Affected Triangles, Max

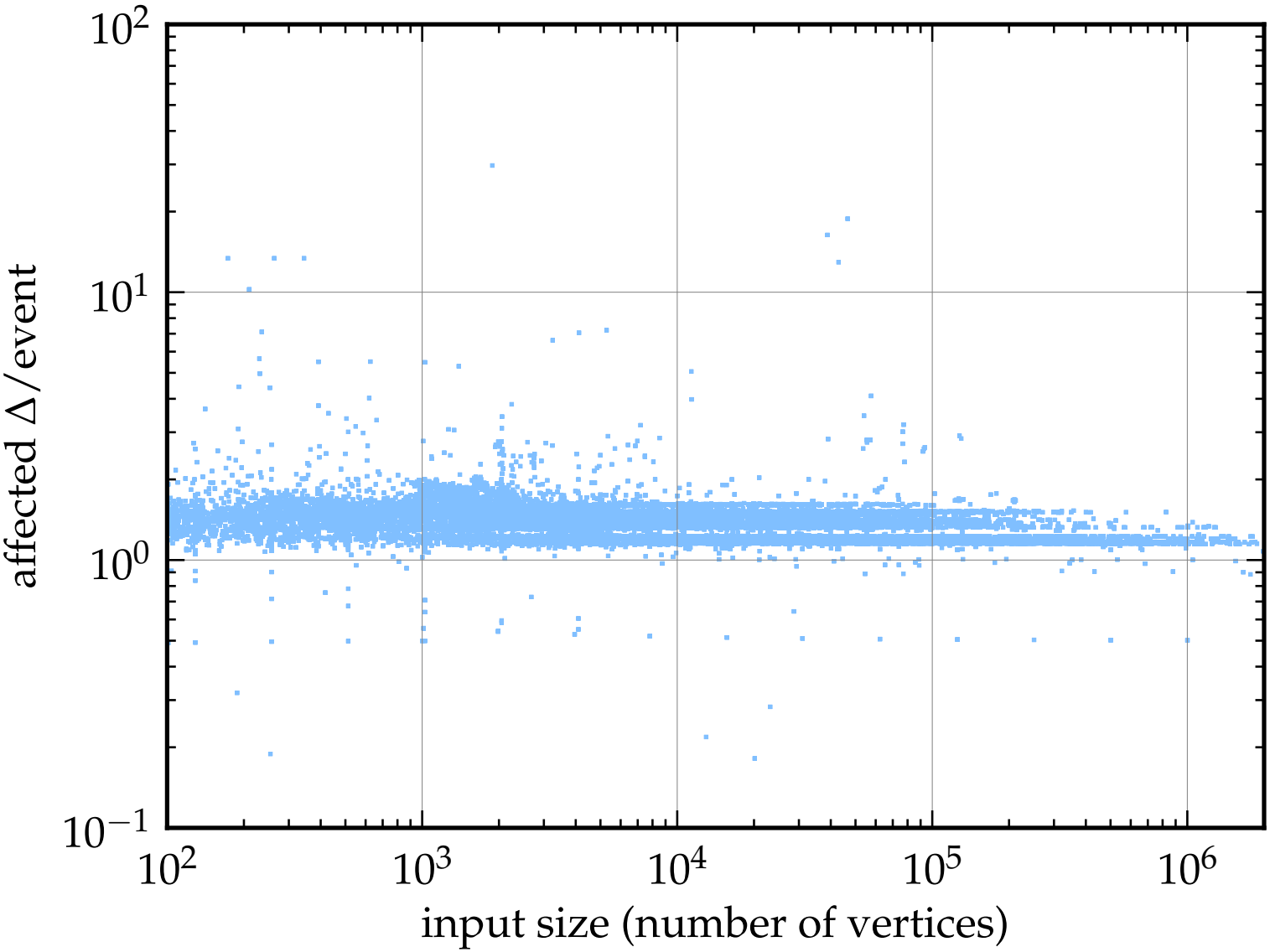


in edge events

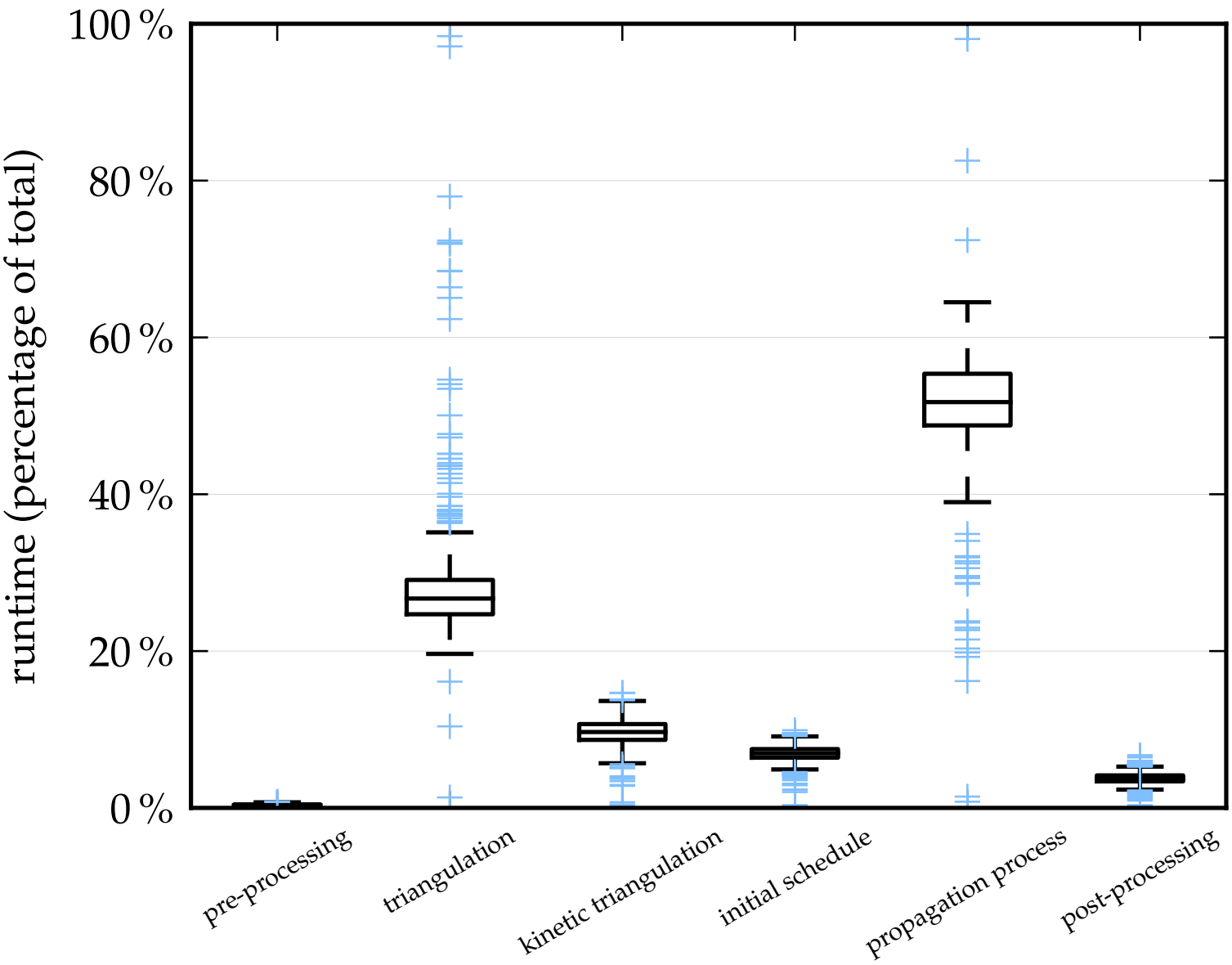


in split events

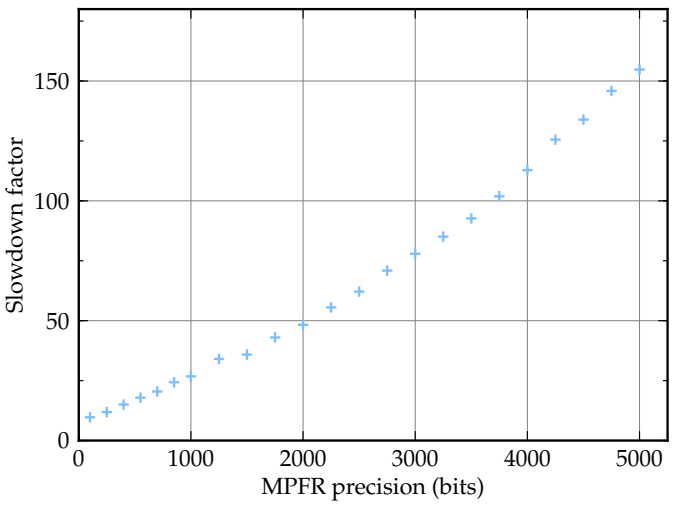
Affected Triangles, Avg



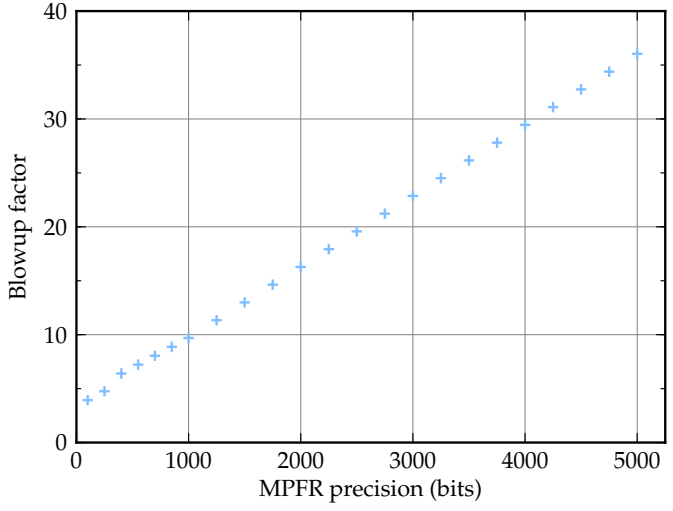
Time spent, phases



MPFR



slowdown



blowup