Computing Mitered Offset Curves Based on Straight Skeletons

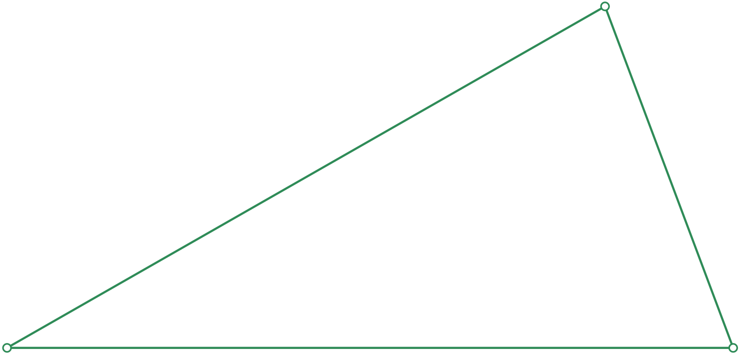
Peter Palfrader Martin Held

Universität Salzburg
FB Computerwissenschaften
Salzburg, Austria

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Straight Skeletons — Motivation

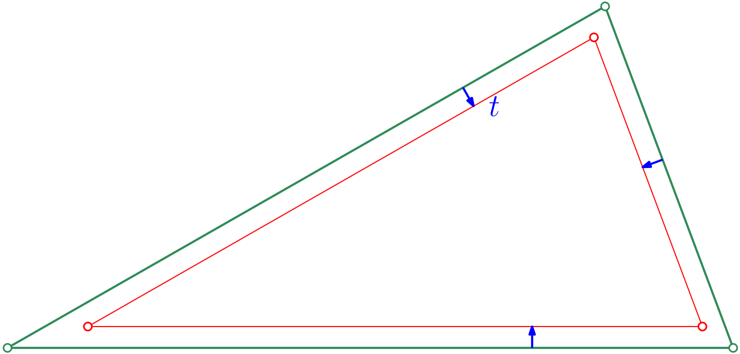
Aichholzer&Alberts&Aurenhammer&Gärtner (1995)



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

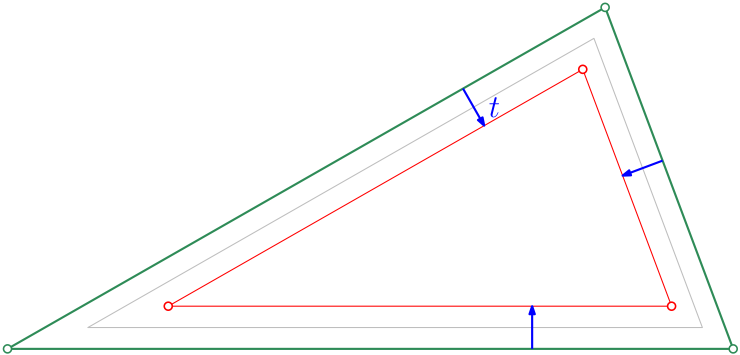
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

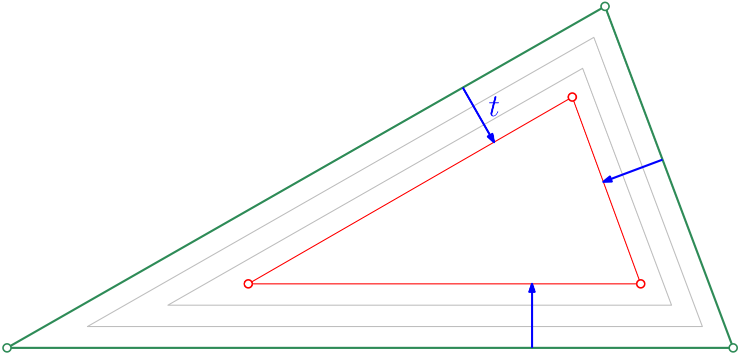
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

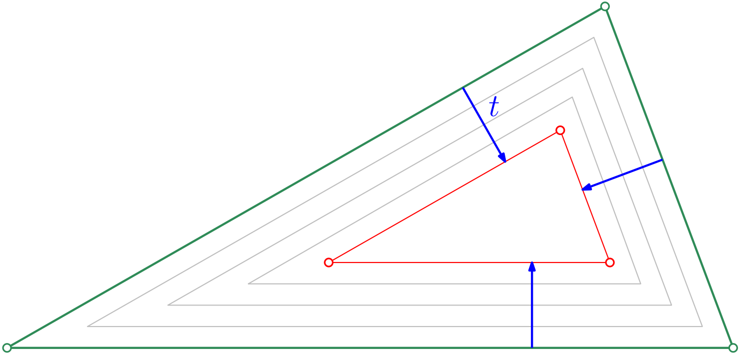
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

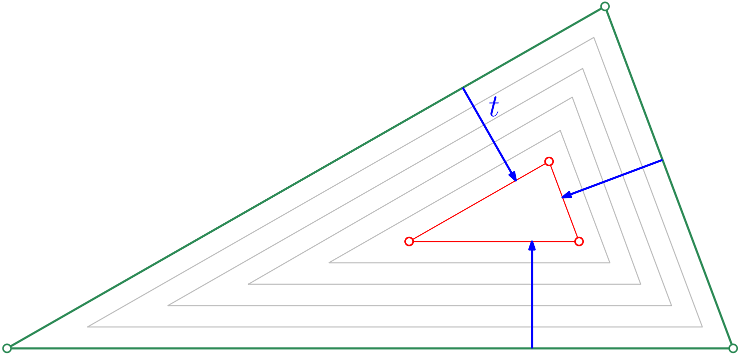
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

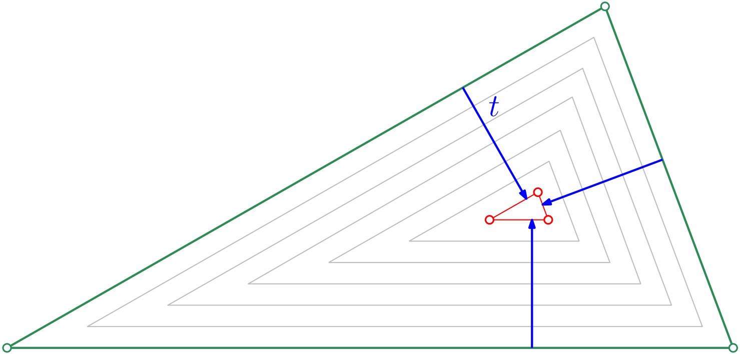
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

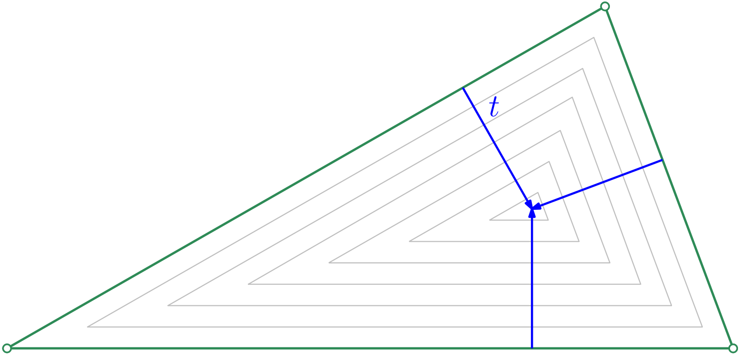
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

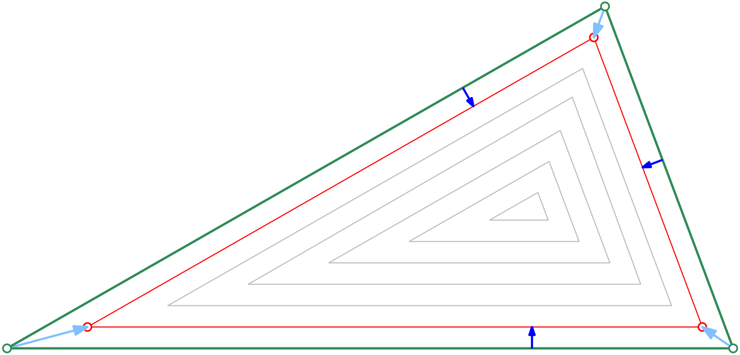
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

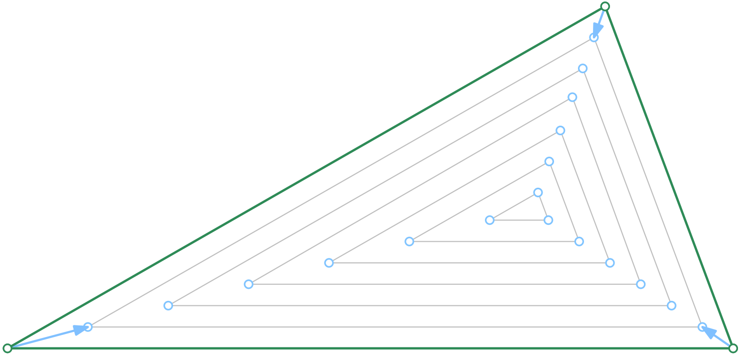
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.



Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.

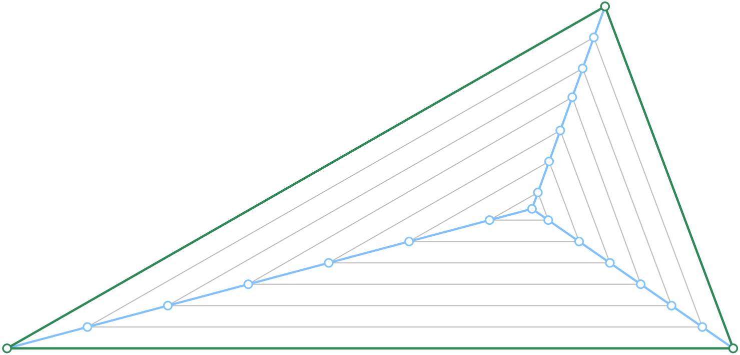


UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Straight Skeletons — Motivation

Aichholzer&Alberts&Aurenhammer&Gärtner (1995)

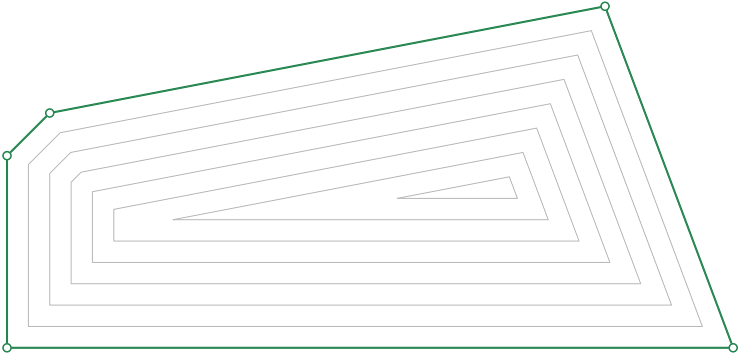
- Offsetting of input polygon $\mathcal{P}$ yields wavefront $\mathcal{WF}(\mathcal{P}, t)$ for offset distance t.
- Wavefront propagation with unit speed via continued offsetting: shrinking process, where offset distance t equals time.
- Straight skeleton $\mathcal{SK}(\mathcal{P})$ is union of traces of wavefront vertices.



Change of Wavefront Topology

Edge event

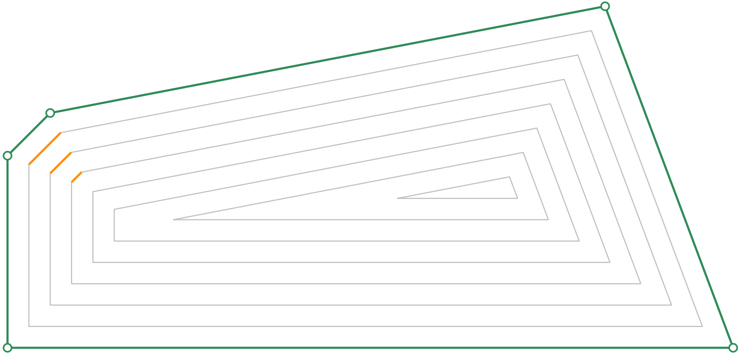
- Wavefront topology changes over time.



Change of Wavefront Topology

Edge event

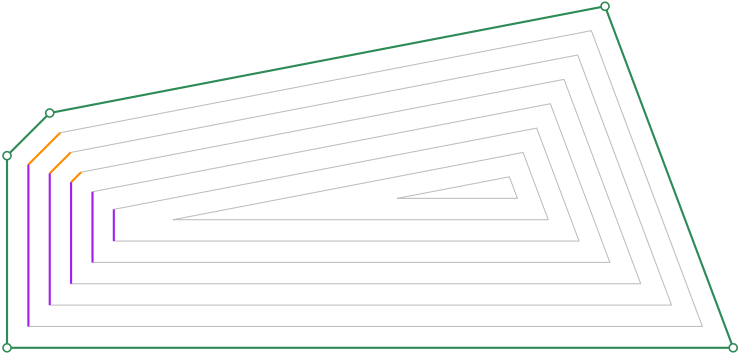
- Wavefront topology changes over time.



Change of Wavefront Topology

Edge event

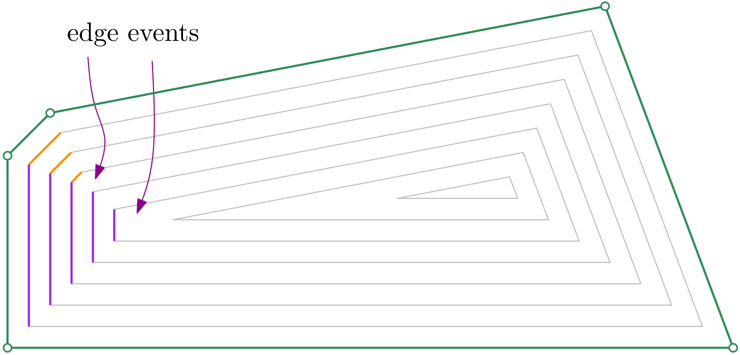
- Wavefront topology changes over time.



Change of Wavefront Topology

Edge event

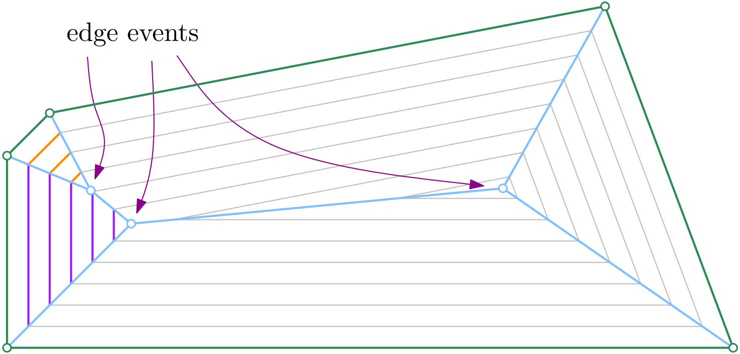
- Wavefront topology changes over time.
- *Edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.



Change of Wavefront Topology

Edge event

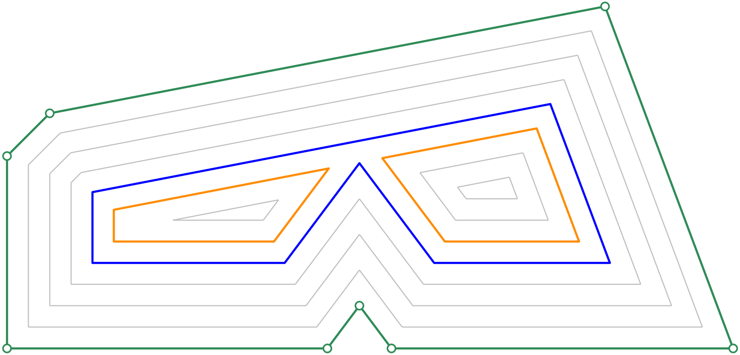
- Wavefront topology changes over time.
- *Edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.
- Such a change of topology corresponds to a *node* of $\mathcal{SK}(\mathcal{P})$.



Change of Wavefront Topology

Split event

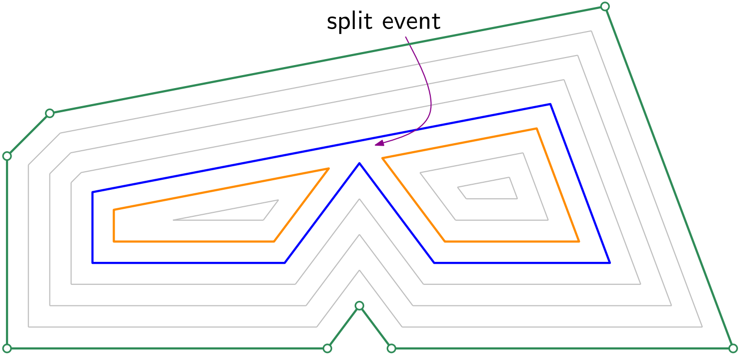
- Wavefront topology changes over time.



Change of Wavefront Topology

Split event

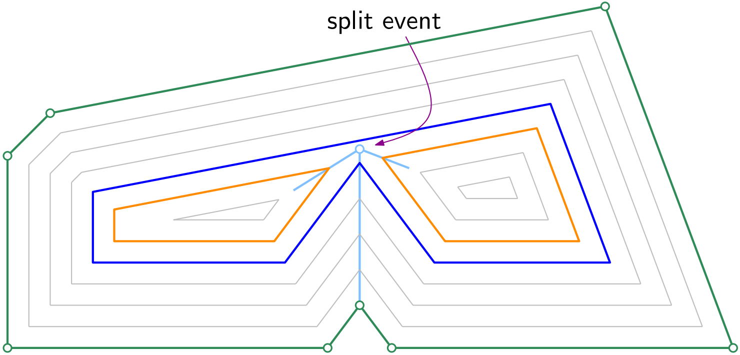
- Wavefront topology changes over time.
- *Split event*: wavefront splits into two parts.



Change of Wavefront Topology

Split event

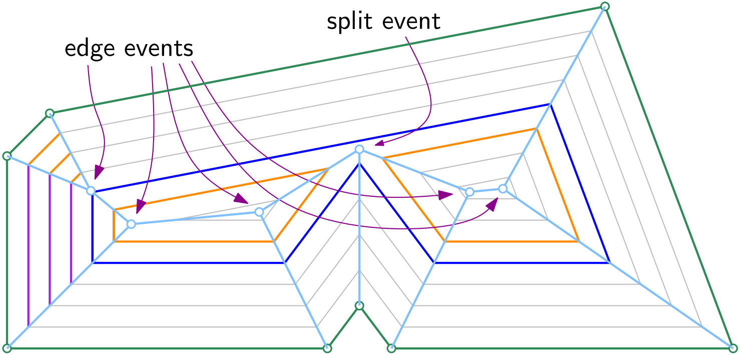
- Wavefront topology changes over time.
- *Split event*: wavefront splits into two parts.
- Also split events correspond to *nodes* of $\mathcal{SK}(\mathcal{P})$.



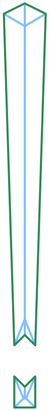
Change of Wavefront Topology

Split event

- Wavefront topology changes over time.
- *Split event*: wavefront splits into two parts.
- Also split events correspond to *nodes* of $\mathcal{SK}(\mathcal{P})$.

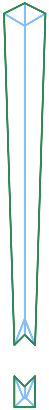


Straight Skeleton

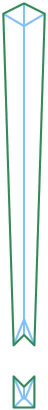


Definition

The *straight skeleton* $\mathcal{SK}(\mathcal{P})$ of a polygon $\mathcal{P}$ is given by the union of traces of wavefront vertices of $\mathcal{P}$ over the entire wavefront propagation process.



Straight Skeleton

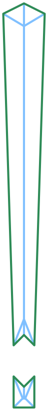


Definition

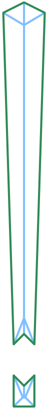
The *straight skeleton* $\mathcal{SK}(\mathcal{P})$ of a polygon $\mathcal{P}$ is given by the union of traces of wavefront vertices of $\mathcal{P}$ over the entire wavefront propagation process.

Basic facts

- The topology of the wavefront $\mathcal{WF}(\mathcal{P}, t)$ changes with time/distance t due to edge and split events.



Straight Skeleton

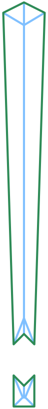


Definition

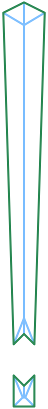
The *straight skeleton* $\mathcal{SK}(\mathcal{P})$ of a polygon $\mathcal{P}$ is given by the union of traces of wavefront vertices of $\mathcal{P}$ over the entire wavefront propagation process.

Basic facts

- The topology of the wavefront $\mathcal{WF}(\mathcal{P}, t)$ changes with time/distance t due to edge and split events.
- These events correspond to nodes of $\mathcal{SK}(\mathcal{P})$.



Straight Skeleton

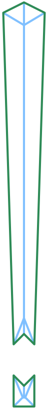


Definition

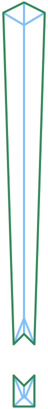
The *straight skeleton* $\mathcal{SK}(\mathcal{P})$ of a polygon $\mathcal{P}$ is given by the union of traces of wavefront vertices of $\mathcal{P}$ over the entire wavefront propagation process.

Basic facts

- The topology of the wavefront $\mathcal{WF}(\mathcal{P}, t)$ changes with time/distance t due to edge and split events.
- These events correspond to nodes of $\mathcal{SK}(\mathcal{P})$.
- No metric-based definition of straight skeletons exists.



Straight Skeleton

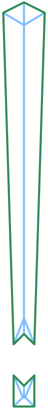


Definition

The *straight skeleton* $\mathcal{SK}(\mathcal{P})$ of a polygon $\mathcal{P}$ is given by the union of traces of wavefront vertices of $\mathcal{P}$ over the entire wavefront propagation process.

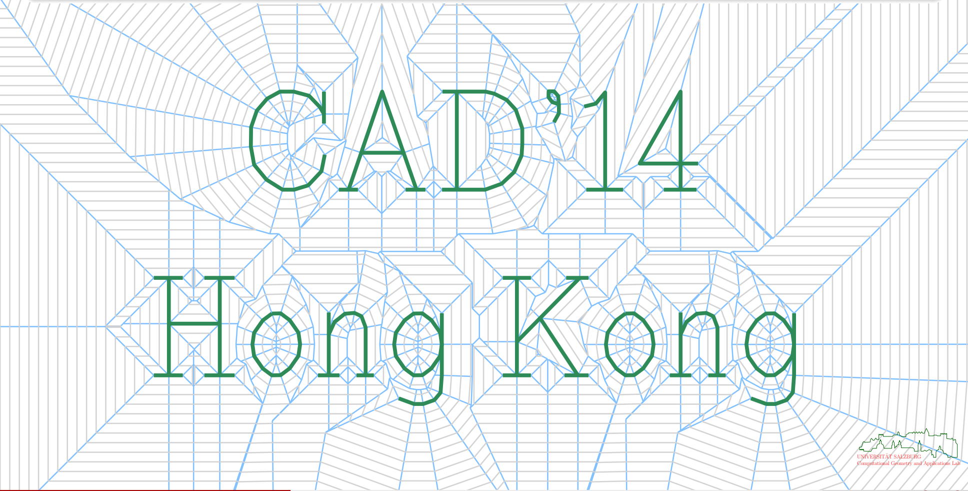
Basic facts

- The topology of the wavefront $\mathcal{WF}(\mathcal{P}, t)$ changes with time/distance t due to edge and split events.
- These events correspond to nodes of $\mathcal{SK}(\mathcal{P})$.
- No metric-based definition of straight skeletons exists.
- If $\mathcal{P}$ has n segments then $\mathcal{SK}(\mathcal{P})$ consists of $O(n)$ nodes and $O(n)$ straight-line edges.

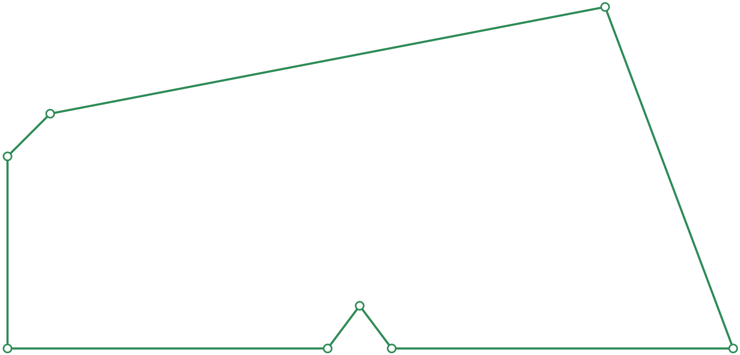


Straight Skeleton of a PSLG

The definition of straight skeletons can be extended easily to arbitrary planar straight line graphs (PSLGs) within the entire plane, i.e., to a collection of straight-line segments that do not intersect except possibly at common endpoints.



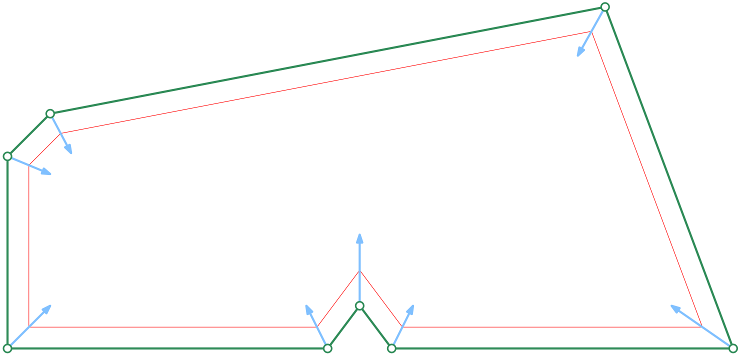
Computing Straight Skeletons



Computing Straight Skeletons

Basic idea

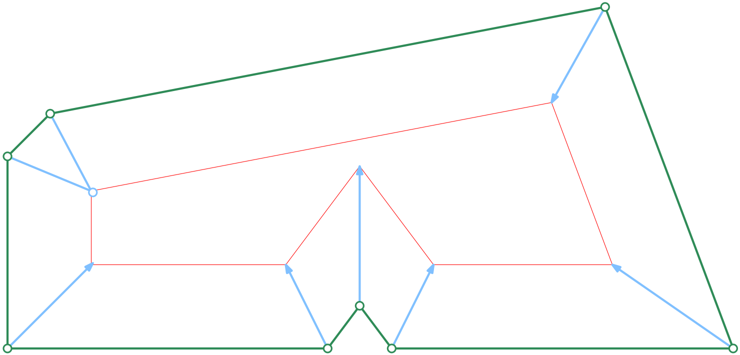
- Simulate the wavefront propagation.



Computing Straight Skeletons

Basic idea

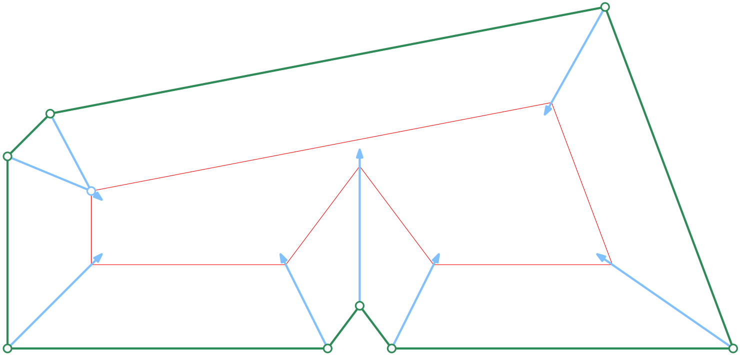
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?



Computing Straight Skeletons

Basic idea

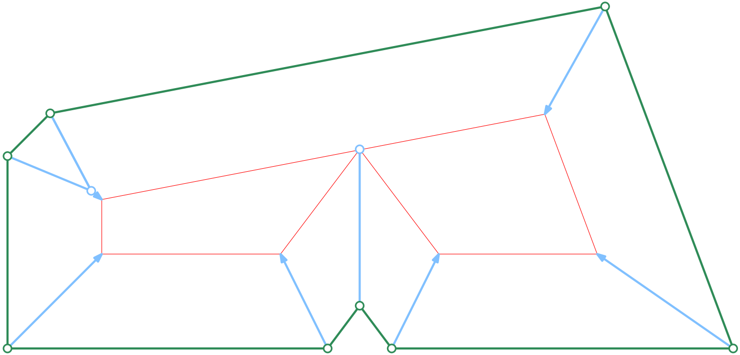
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.



Computing Straight Skeletons

Basic idea

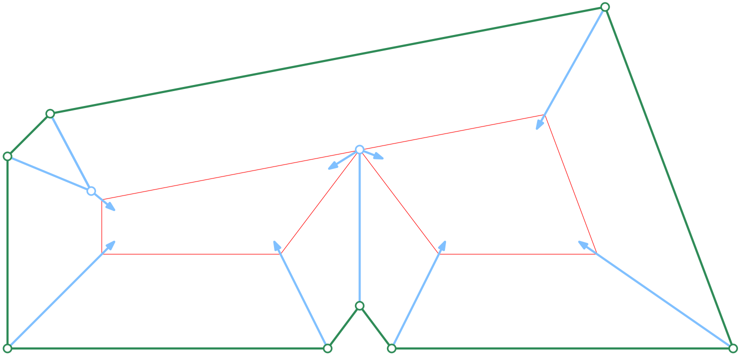
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.



Computing Straight Skeletons

Basic idea

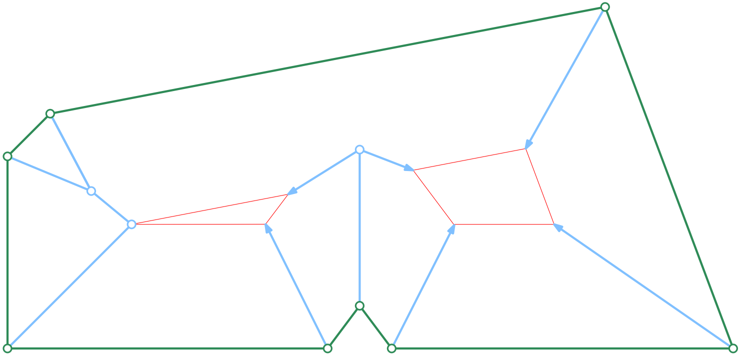
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.



Computing Straight Skeletons

Basic idea

- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.

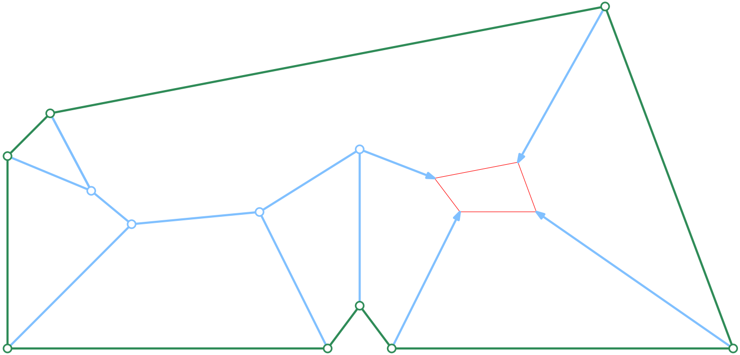


UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Computing Straight Skeletons

Basic idea

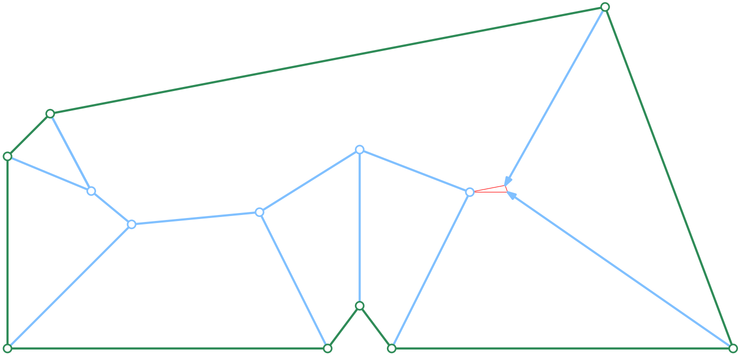
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.



Computing Straight Skeletons

Basic idea

- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.

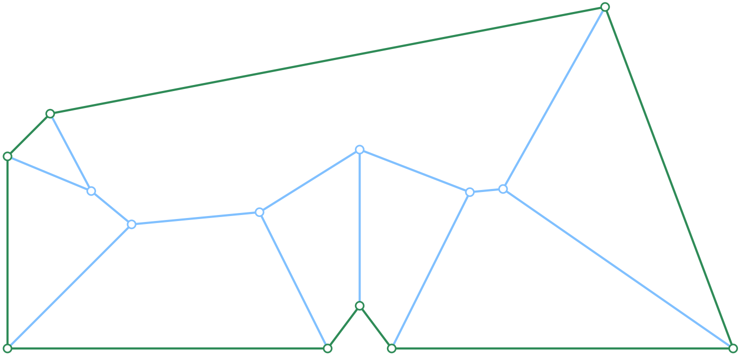


UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Computing Straight Skeletons

Basic idea

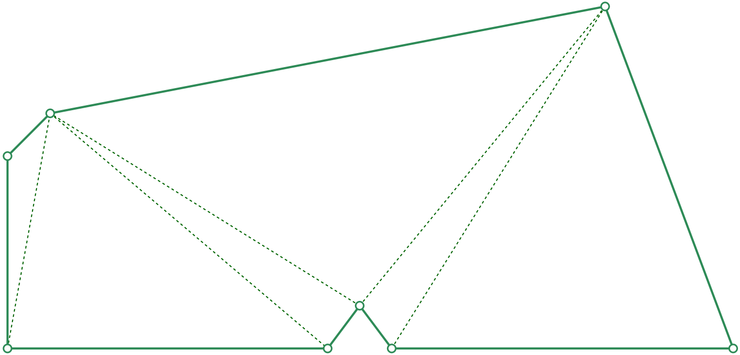
- Simulate the wavefront propagation.
- Problem: When will the next event happen? Which event?
- If we can solve this problem then we can construct straight skeletons.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

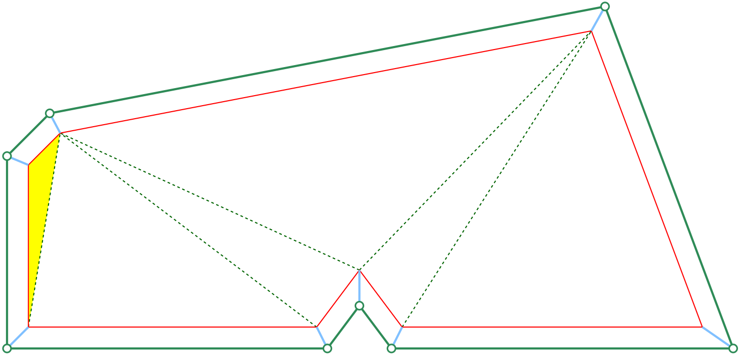
- Maintain a kinetic triangulation of (the interior of) the wavefront.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

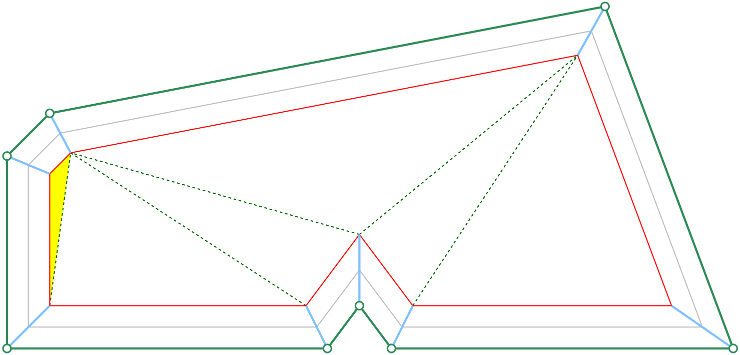
- Maintain a kinetic triangulation of (the interior of) the wavefront.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

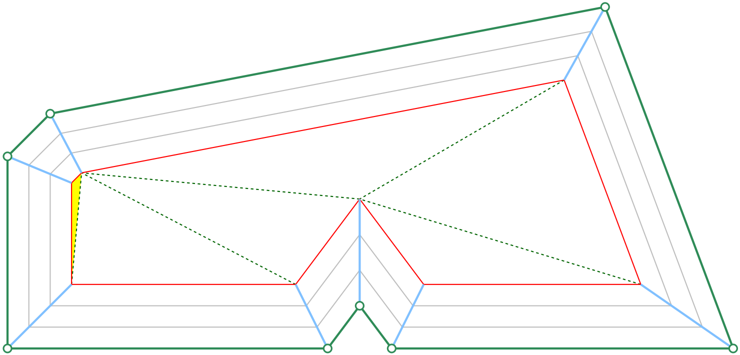
- Maintain a kinetic triangulation of (the interior of) the wavefront.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

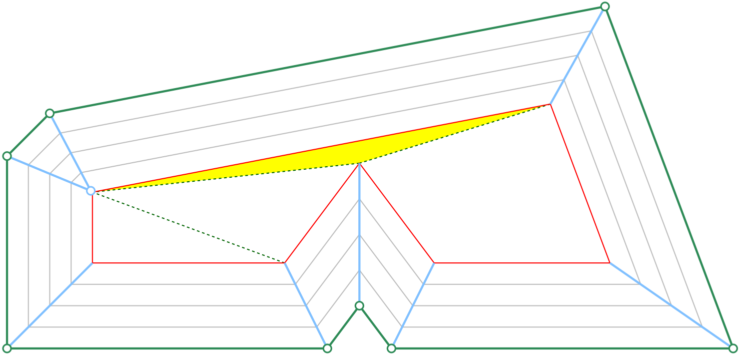
- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.

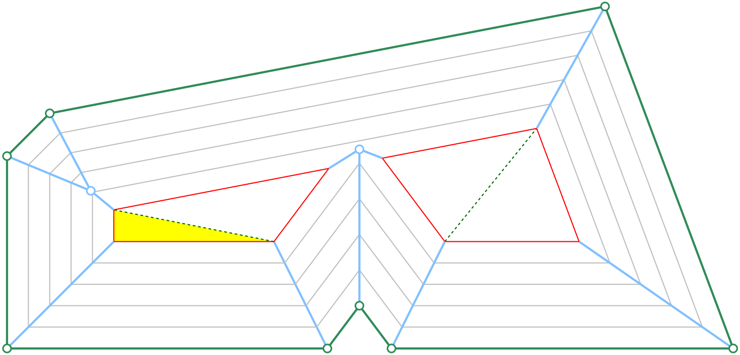


UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

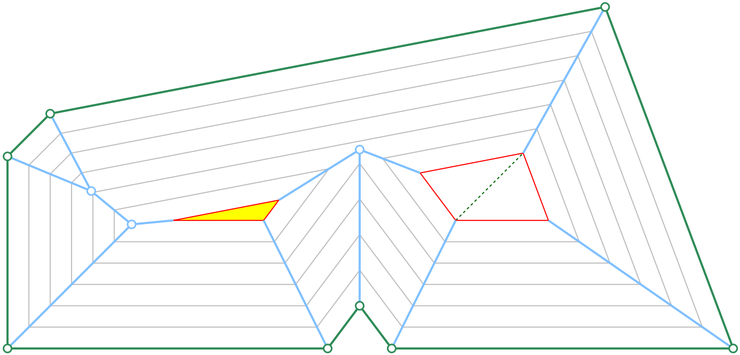
- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.
- A triangle collapses when its area becomes zero.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

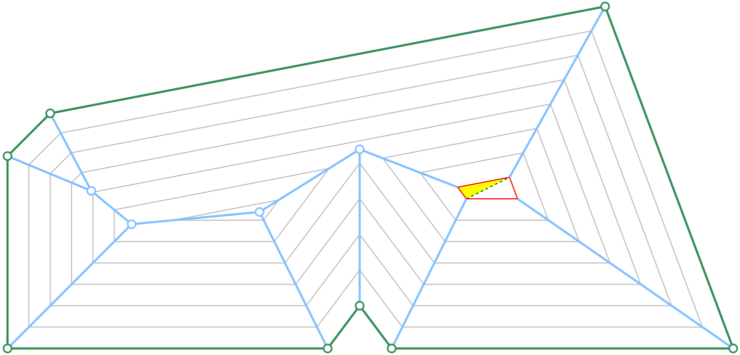
- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.
- A triangle collapses when its area becomes zero.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

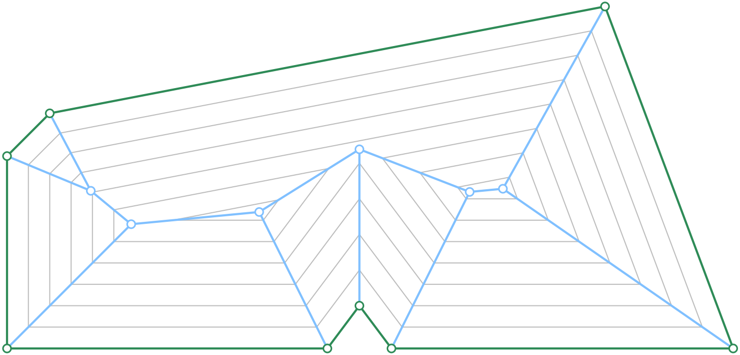
- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.
- A triangle collapses when its area becomes zero.



Triangulation-Based Algorithm

Aichholzer&Aurenhammer (1998)

- Maintain a kinetic triangulation of (the interior of) the wavefront.
- Collapsing triangles witness edge and split events.
- A triangle collapses when its area becomes zero.

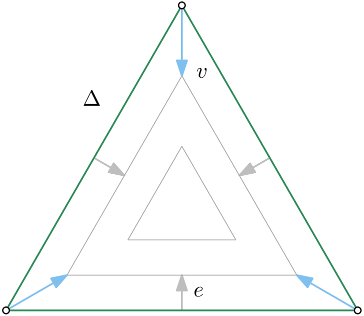


Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.

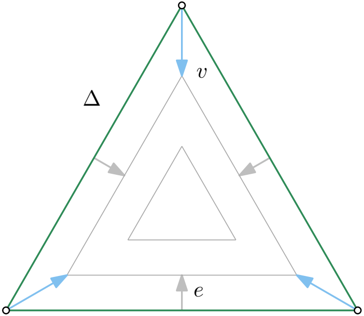
Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.
- Compute collapse times of triangles.



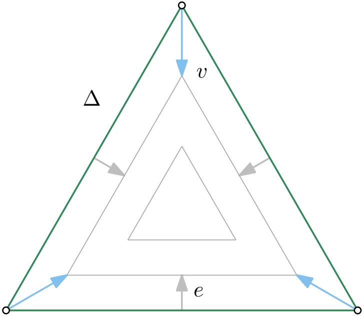
Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.
- Compute collapse times of triangles.
- That is, determine when the area of a triangle becomes zero.



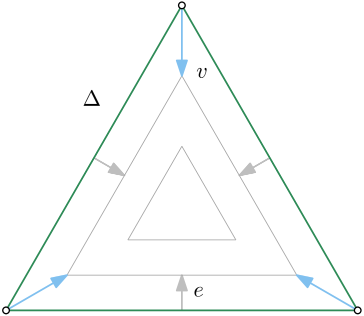
Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.
- Compute collapse times of triangles.
- That is, determine when the area of a triangle becomes zero.
- Maintain a priority queue of collapse events.



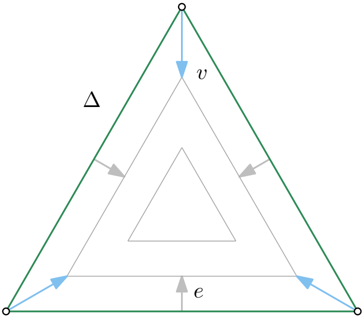
Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.
- Compute collapse times of triangles.
- That is, determine when the area of a triangle becomes zero.
- Maintain a priority queue of collapse events.
- Update triangulation and priority queue as required upon events.



Triangulation-Based Algorithm

- Collapsing triangles witness edge and split events.
- Compute collapse times of triangles.
- That is, determine when the area of a triangle becomes zero.
- Maintain a priority queue of collapse events.
- Update triangulation and priority queue as required upon events.



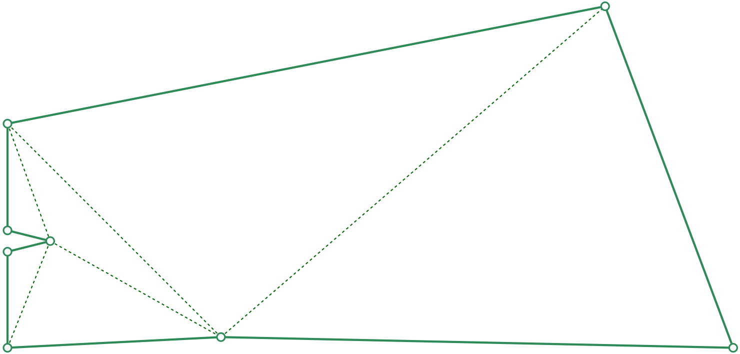
Algorithmic insight

Wavefront propagation based on kinetic triangulations allows to determine all events and to compute straight skeletons.

Triangulation-based Algorithm

Flip events

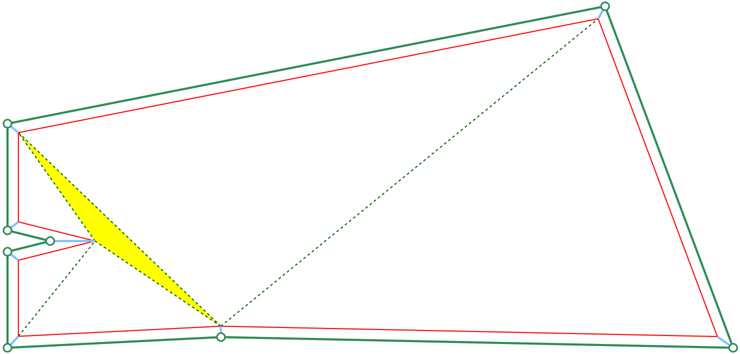
- Caveat: Not all collapses witness changes in the wavefront topology.



Triangulation-based Algorithm

Flip events

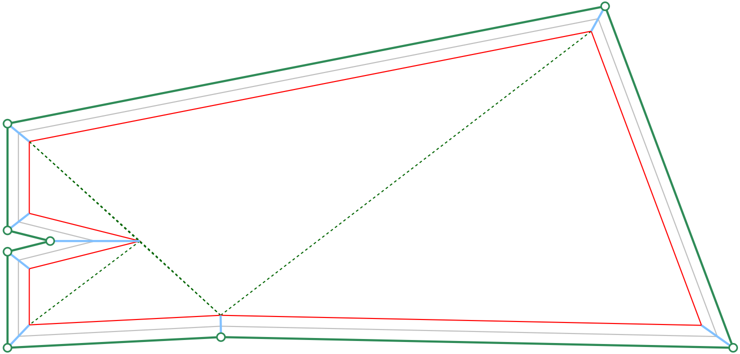
- Caveat: Not all collapses witness changes in the wavefront topology.



Triangulation-based Algorithm

Flip events

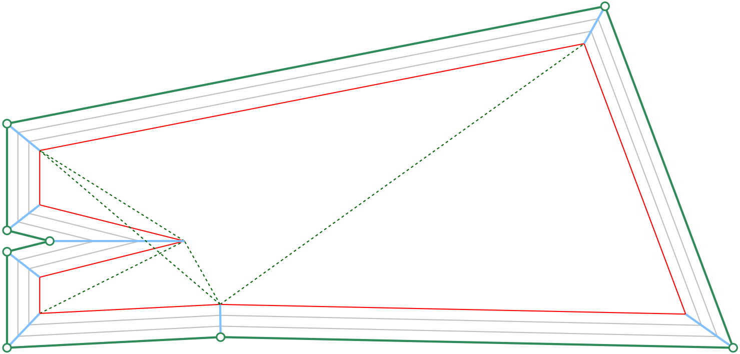
- Caveat: Not all collapses witness changes in the wavefront topology.



Triangulation-based Algorithm

Flip events

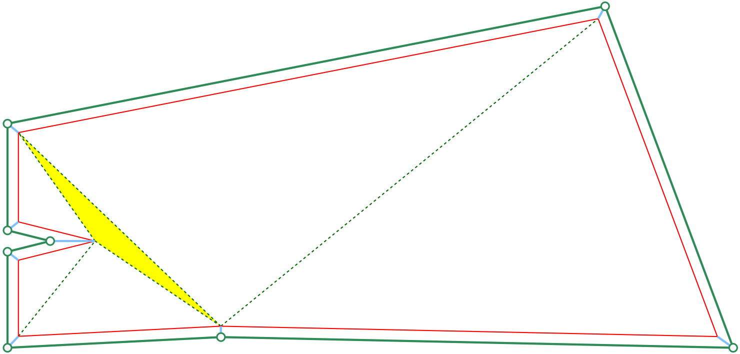
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!



Triangulation-based Algorithm

Flip events

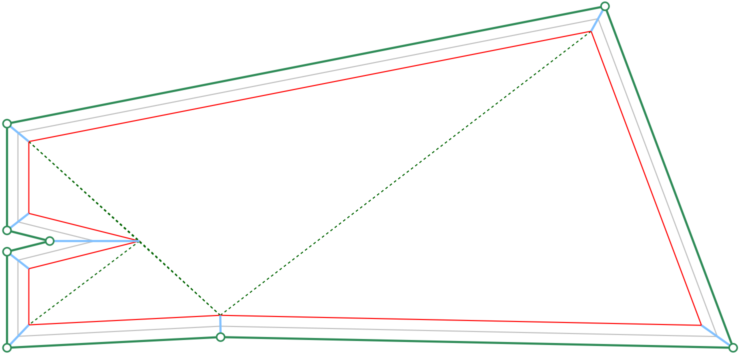
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Triangulation-based Algorithm

Flip events

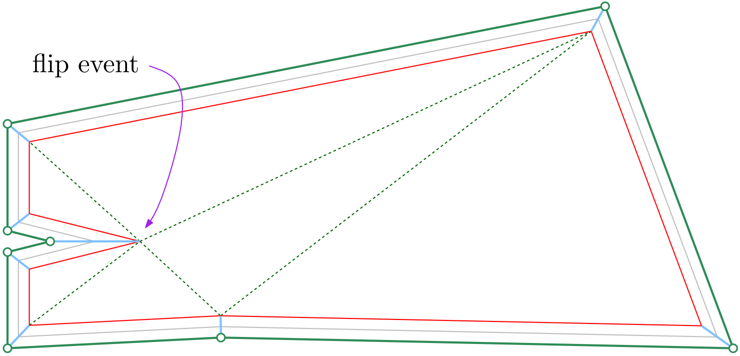
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Triangulation-based Algorithm

Flip events

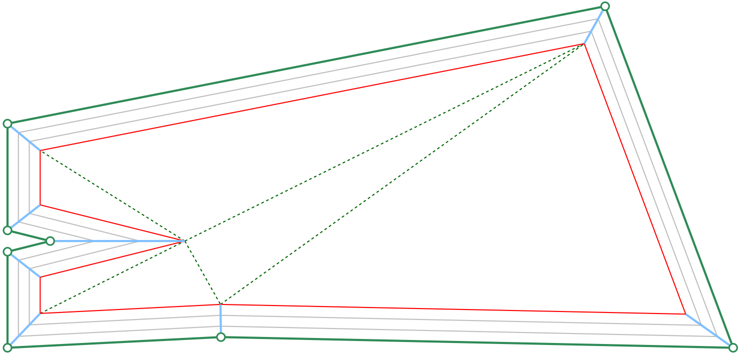
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Triangulation-based Algorithm

Flip events

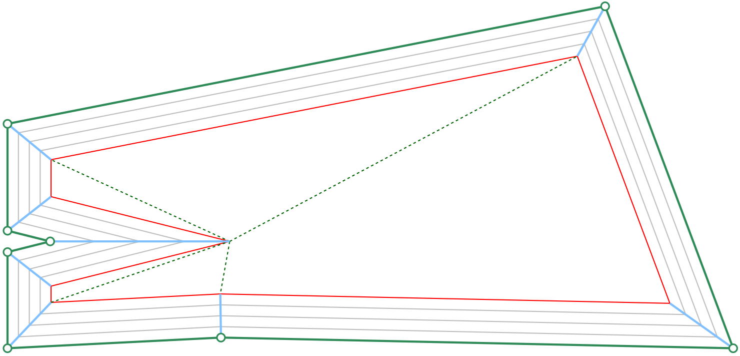
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Triangulation-based Algorithm

Flip events

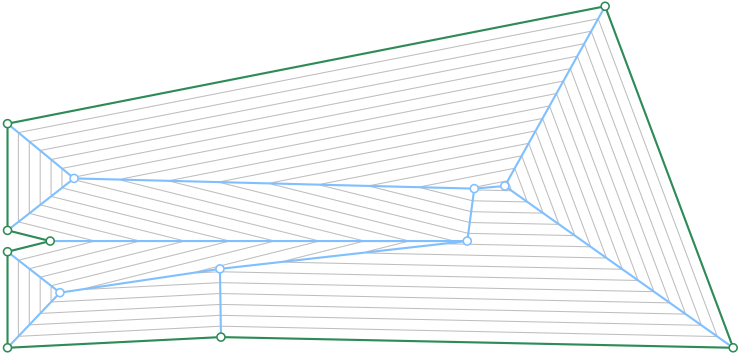
- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Triangulation-based Algorithm

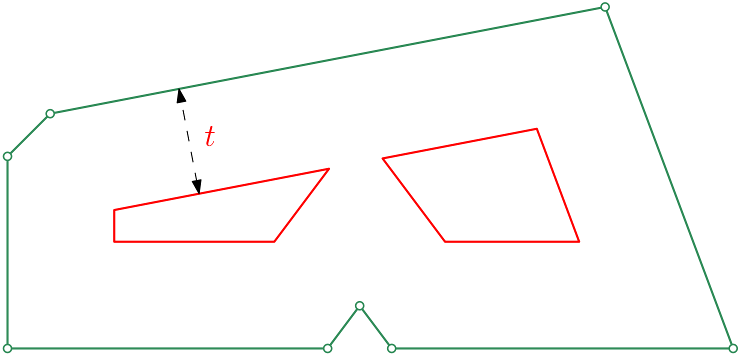
Flip events

- Caveat: Not all collapses witness changes in the wavefront topology.
- Such collapses cannot be ignored!
- Rather these collapes need special processing: *flip events*.



Offsetting

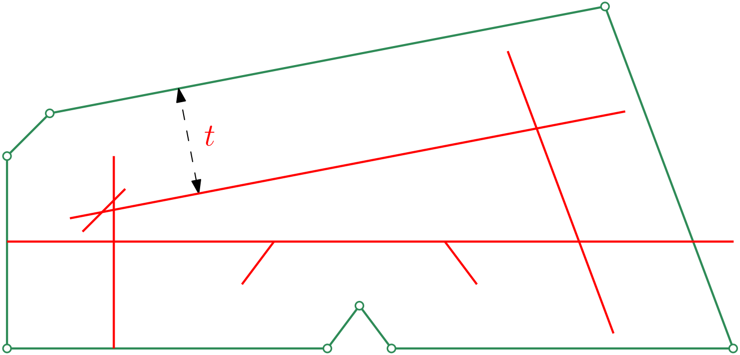
- How can we determine all offsets that correspond to some user-specified offset distance t?



Offsetting

Standard approach

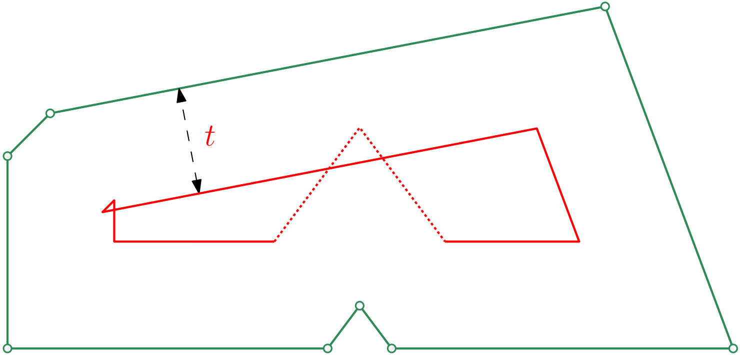
1. Compute an elementary offset segment for each input segment.



Offsetting

Standard approach

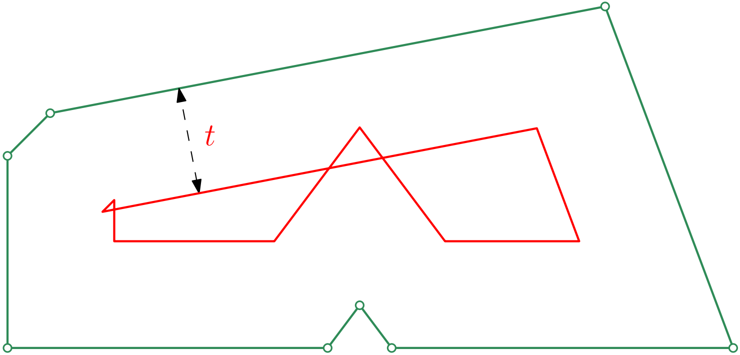
1. Compute an elementary offset segment for each input segment.
2. Trim at intersections of neighboring segments,



Offsetting

Standard approach

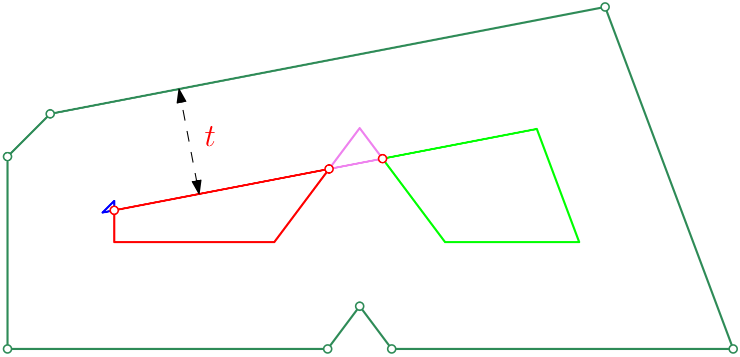
1. Compute an elementary offset segment for each input segment.
2. Trim at intersections of neighboring segments, and close gaps to form one loop.



Offsetting

Standard approach

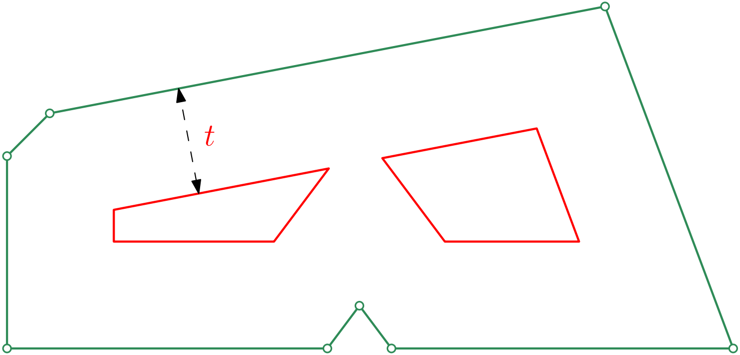
1. Compute an elementary offset segment for each input segment.
2. Trim at intersections of neighboring segments, and close gaps to form one loop.
3. Determine all self-intersections and split into several loops.



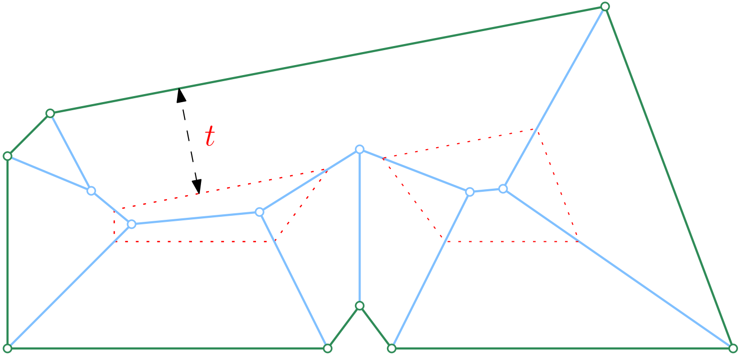
Offsetting

Standard approach

1. Compute an elementary offset segment for each input segment.
2. Trim at intersections of neighboring segments, and close gaps to form one loop.
3. Determine all self-intersections and split into several loops.
4. Discard excess loops.



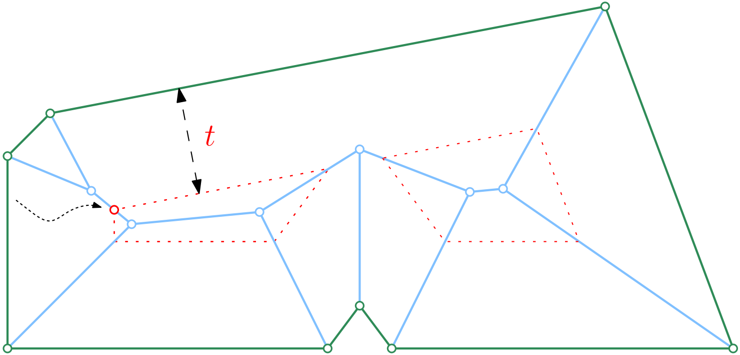
Offsetting Based on Straight Skeleton



Offsetting Based on Straight Skeleton

Scan straight skeleton

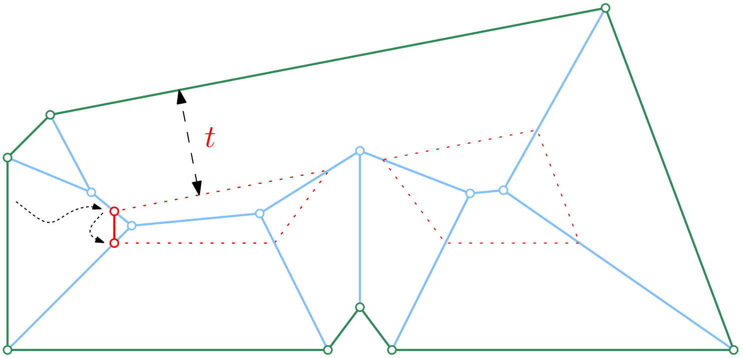
1. Choose SK edge not yet intersected by an offset loop; compute start vertex.



Offsetting Based on Straight Skeleton

Scan straight skeleton

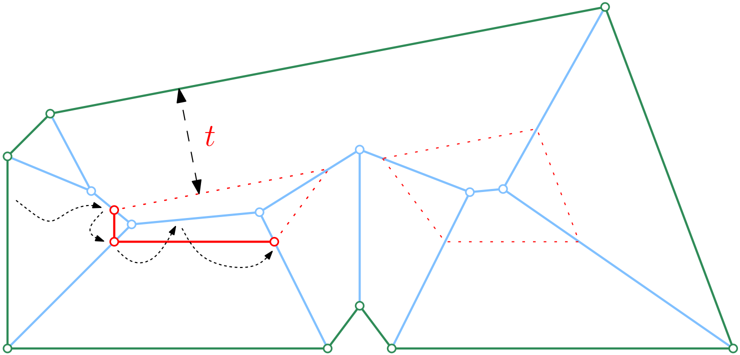
1. Choose SK edge not yet intersected by an offset loop; compute start vertex.
2. Advance clockwise along boundary of SK face and compute next vertex.



Offsetting Based on Straight Skeleton

Scan straight skeleton

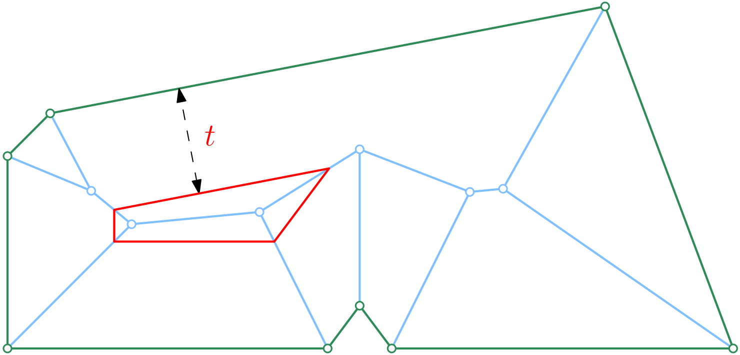
1. Choose SK edge not yet intersected by an offset loop; compute start vertex.
2. Advance clockwise along boundary of SK face and compute next vertex.
3. Move to neighboring face and keep scanning that face clockwise.



Offsetting Based on Straight Skeleton

Scan straight skeleton

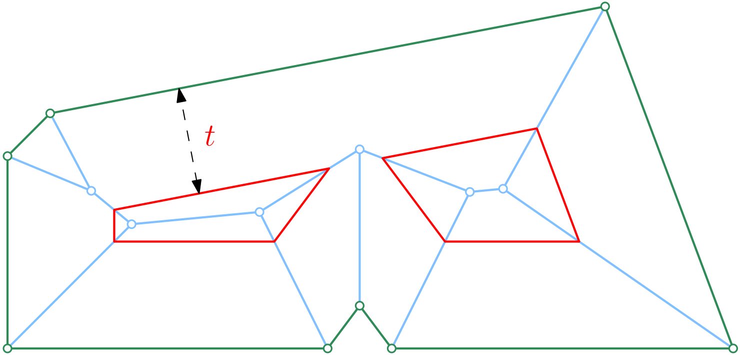
1. Choose SK edge not yet intersected by an offset loop; compute start vertex.
2. Advance clockwise along boundary of SK face and compute next vertex.
3. Move to neighboring face and keep scanning that face clockwise.
4. Finish one offset curve.



Offsetting Based on Straight Skeleton

Scan straight skeleton

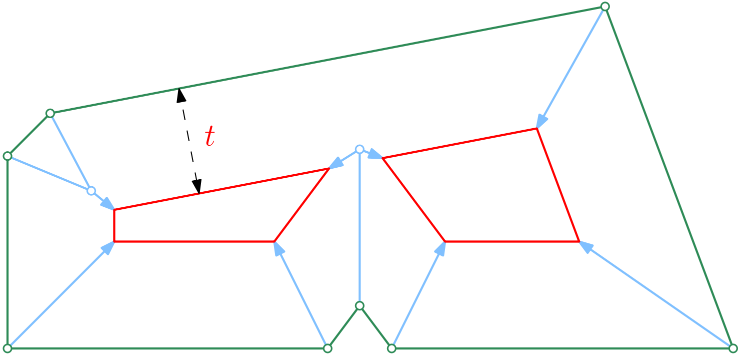
1. Choose SK edge not yet intersected by an offset loop; compute start vertex.
2. Advance clockwise along boundary of SK face and compute next vertex.
3. Move to neighboring face and keep scanning that face clockwise.
4. Finish one offset curve. Continue with next offset curve.



Offsetting Based on Straight Skeleton

Alternative: Halt wavefront

Halt wavefront-propagation when the offset distance t is reached.



Implementation

Implementation

- Long way to go from the theoretical sketch of Aichholzer&Aurenhammer (1998) to an actual implementation . . .

Implementation

- Long way to go from the theoretical sketch of Aichholzer&Aurenhammer (1998) to an actual implementation . . .
- Need to avoid flip-event loops [Palfrader&Held&Huber (2012)].

Implementation

- Long way to go from the theoretical sketch of Aichholzer&Aurenhammer (1998) to an actual implementation . . .
- Need to avoid flip-event loops [Palfrader&Held&Huber (2012)].
- Need to handle degeneracies that cause multiple simultaneous events [P.&H.].
- Need to detect and classify simultaneous events reliably on a standard floating-point arithmetic [P.&H.].

Implementation

- Long way to go from the theoretical sketch of Aichholzer&Aurenhammer (1998) to an actual implementation . . .
- Need to avoid flip-event loops [Palfrader&Held&Huber (2012)].
- Need to handle degeneracies that cause multiple simultaneous events [P.&H.].
- Need to detect and classify simultaneous events reliably on a standard floating-point arithmetic [P.&H.].

SURFER

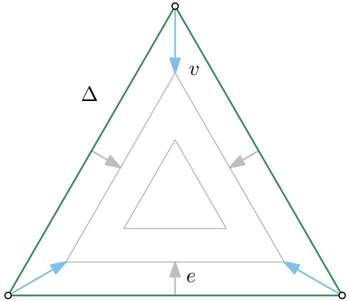
Straight-skeleton algorithm, based on kinetic triangulations, implemented in C and named SURFER.

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Implementation: Finding and Classifying Collapse Times

Different ways to compute collapse time

Suppose that the three vertices of a triangle move towards one point.

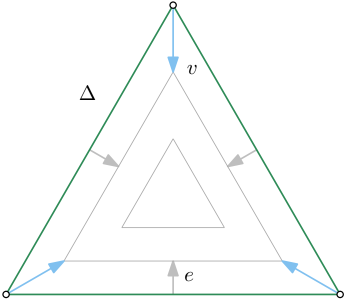


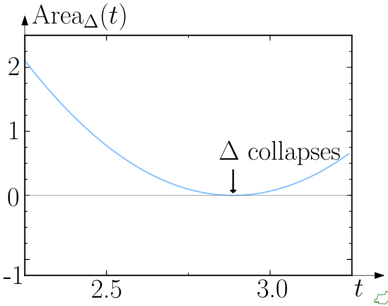
Implementation: Finding and Classifying Collapse Times

Different ways to compute collapse time

Suppose that the three vertices of a triangle move towards one point.

- The parabola plotted in blue is the (signed) area of the triangle over time, e.g., as obtained by means of determinant computations.



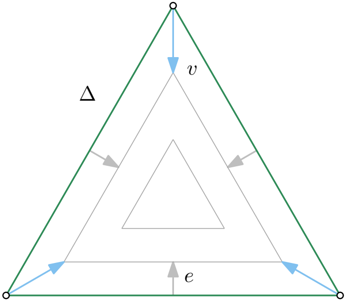


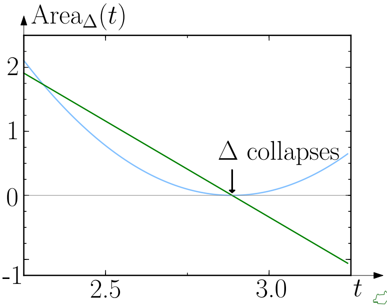
Implementation: Finding and Classifying Collapse Times

Different ways to compute collapse time

Suppose that the three vertices of a triangle move towards one point.

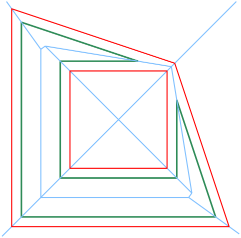
- The parabola plotted in blue is the (signed) area of the triangle over time, e.g., as obtained by means of determinant computations.
- The function in green represents the (signed) distance of one vertex to its opposite edge.





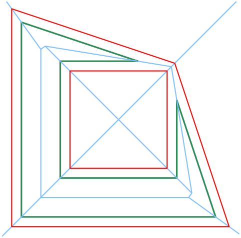
Experiments

- Several straight-skeleton codes:
  - Felkel&Obdržálek (1998),
  - Cacciola/CGAL (2004),
  - Huber&Held ("BONE", 2010).



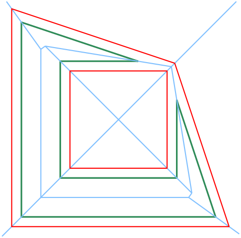
Experiments

- Several straight-skeleton codes:
  - Felkel&Obdržálek (1998),
  - Cacciola/CGAL (2004),
  - Huber&Held ("BONE", 2010).
- SURFER is faster than BONE, which in turn is significantly faster than Cacciola's CGAL code [Palfrader&Held&Huber (2012)].



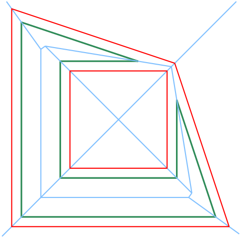
Experiments

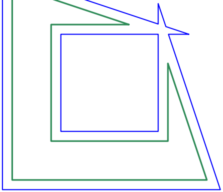
- Several straight-skeleton codes:
  - Felkel&Obdržálek (1998),
  - Cacciola/CGAL (2004),
  - Huber&Held ("BONE", 2010).
- SURFER is faster than BONE, which in turn is significantly faster than Cacciola's CGAL code [Palfrader&Held&Huber (2012)].
- No published/non-proprietary codes dedicated to mitered offsetting are known.



Experiments

- Several straight-skeleton codes:
  - Felkel&Obdržálek (1998),
  - Cacciola/CGAL (2004),
  - Huber&Held ("BONE", 2010).
- SURFER is faster than BONE, which in turn is significantly faster than Cacciola's CGAL code [Palfrader&Held&Huber (2012)].
- No published/non-proprietary codes dedicated to mitered offsetting are known.
- CLIPPER and GEOS: Polygon-clipping libraries that apply general-purpose Boolean clipping algorithms to compute offsets.



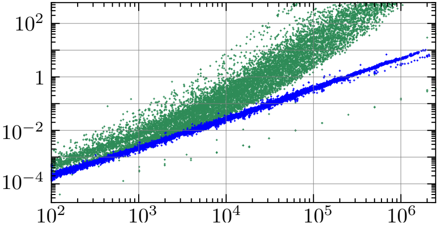


Experiments

- Simple closed polygons as test data.
- Input complexity *n* on *x*-axis, running time in seconds on *y*-axis.

Computation of one offset

- CLIPPER,
- SURFER.

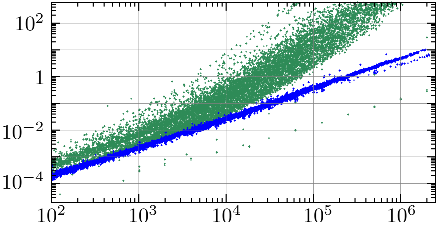


Experiments

- Simple closed polygons as test data.
- Input complexity *n* on *x*-axis, running time in seconds on *y*-axis.

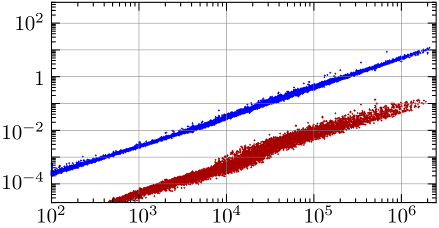
Computation of one offset

- CLIPPER,
- SURFER.



SK and SK-based offsetting

- Full SK by SURFER,
- One offset based on SK.

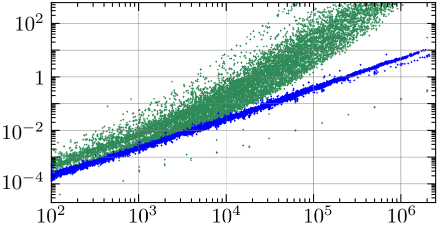


Experiments

- Simple closed polygons as test data.
- Input complexity *n* on *x*-axis, running time in seconds on *y*-axis.

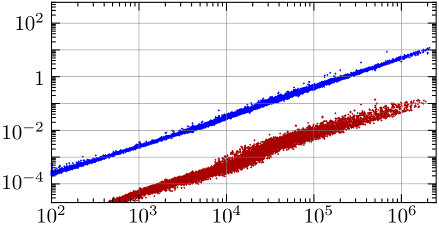
Computation of one offset

- CLIPPER,
- SURFER.



SK and SK-based offsetting

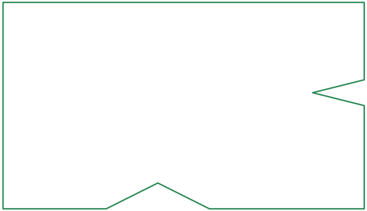
- Full SK by SURFER,
- One offset based on SK.



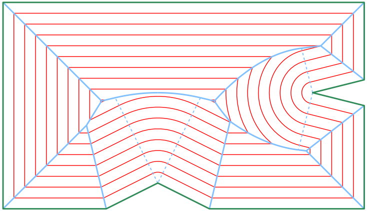
Experimental result

SURFER consumes roughly $5.8 \cdot 10^{-7} n \log n$ microseconds for an *n*-segment input. Except for a few convex polygons, a full run of SURFER is always (substantially) faster than the computation of one offset by CLIPPER.

Comparison of Sample Offsets

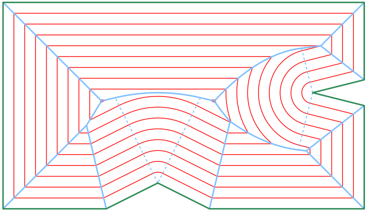


Comparison of Sample Offsets

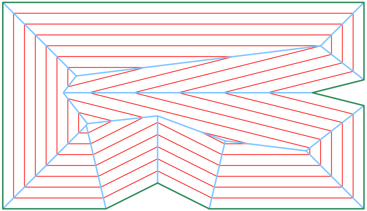


Voronoi diagram and rounded offsets

Comparison of Sample Offsets

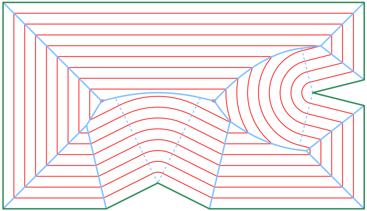


Voronoi diagram and rounded offsets

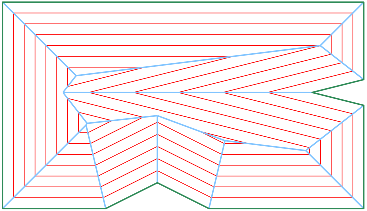


Straight skeleton and mitered offsets

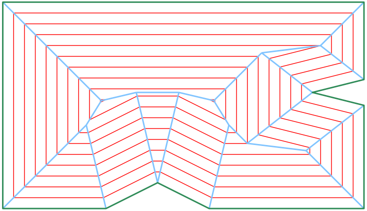
Comparison of Sample Offsets



Voronoi diagram and rounded offsets

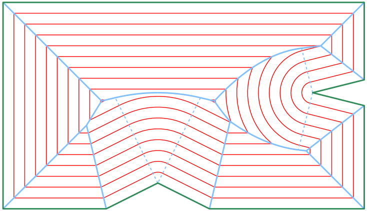


Straight skeleton and mitered offsets

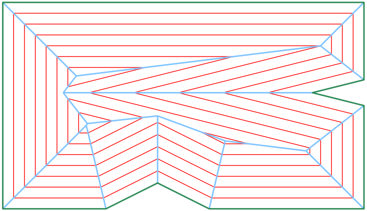


Straight skeleton and beveled offsets

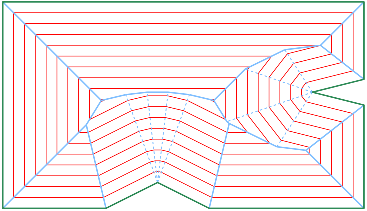
Comparison of Sample Offsets



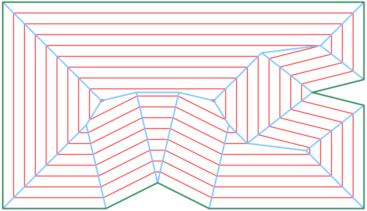
Voronoi diagram and rounded offsets



Straight skeleton and mitered offsets

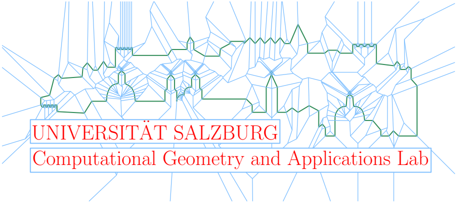


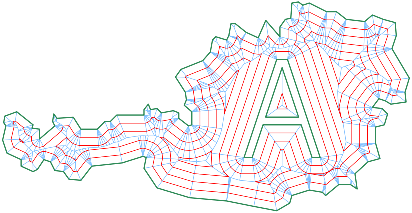
Linear axis and multi-segment bevels

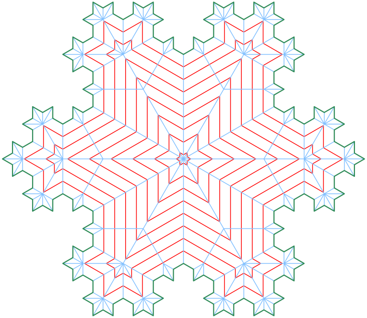


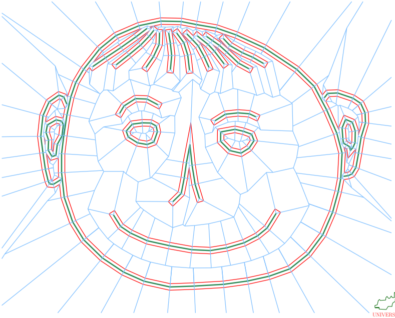
Straight skeleton and beveled offsets

Gallery


UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab







Thanks for Your Attention — Questions Welcome!

1. **Introduction**
   - Motivation
   - Change of Wavefront Topology
   - Definition of Straight Skeleton
2. **Triangulation-Based Algorithm**
   - Basic Idea
   - Kinetic Triangulation
3. **Offsetting**
   - Standard Approach
   - Offsetting Based on SK
4. **Implementation**
5. **Experimental Results**
6. **Gallery**

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab