# On Computing Straight Skeletons by Means of Kinetic Triangulations

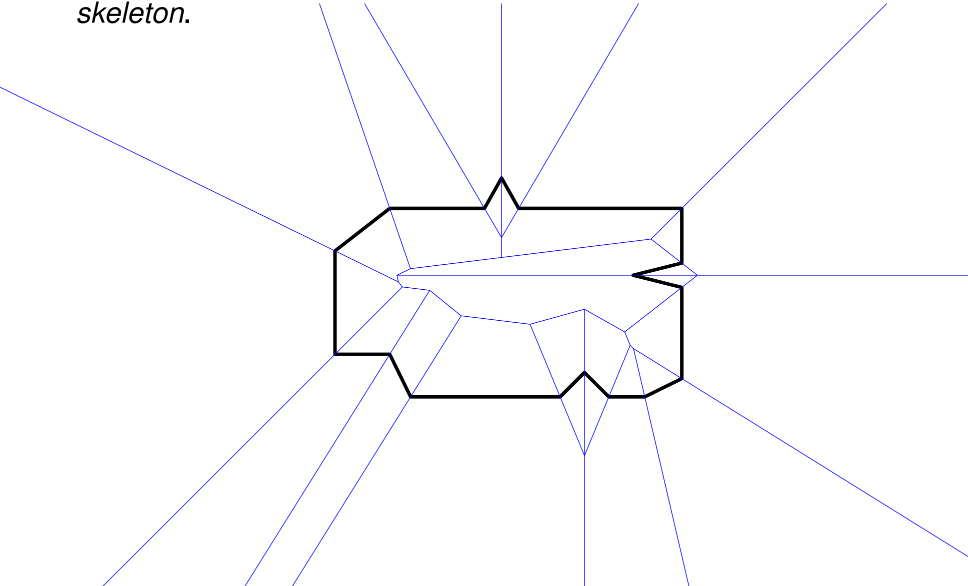
Peter Palfrader Martin Held Stefan Huber

University of Salzburg

European Symposium on Algorithms 2012

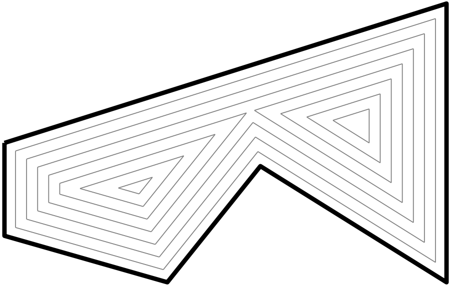
## Problem

Given a planar, straight line graph, construct the *straight skeleton*.



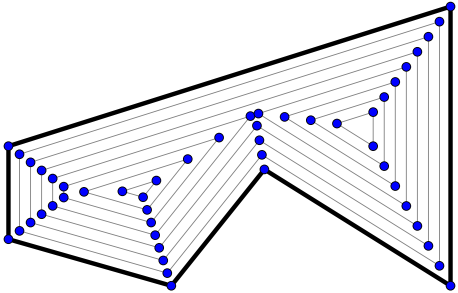
# Defining Straight Skeletons

- Introduced by Aichholzer et al. in 1995.
- A skeleton consisting exclusively of straight line segments.
- Defined by a *wavefront propagation process*: The straight skeleton is the set of loci that are traced out by wavefront vertices.



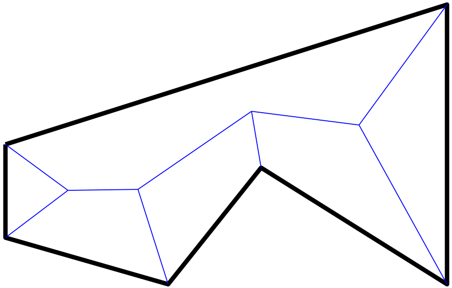
# Defining Straight Skeletons

- Introduced by Aichholzer et al. in 1995.
- A skeleton consisting exclusively of straight line segments.
- Defined by a *wavefront propagation process*: The straight skeleton is the set of loci that are traced out by wavefront vertices.



# Defining Straight Skeletons

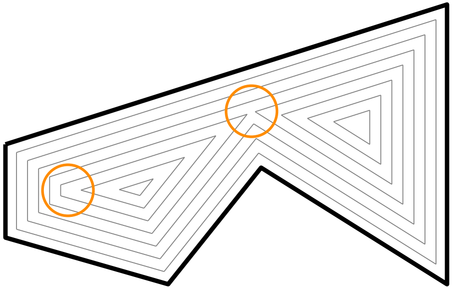
- Introduced by Aichholzer et al. in 1995.
- A skeleton consisting exclusively of straight line segments.
- Defined by a *wavefront propagation process*: The straight skeleton is the set of loci that are traced out by wavefront vertices.



# Wavefront propagation

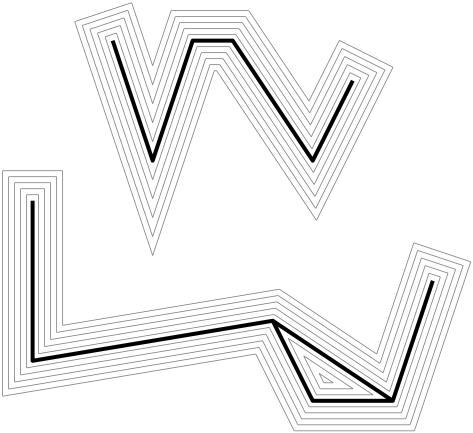
During this wavefront propagation, the wavefront topology changes:

- Split events: a wavefront vertex crashes into an oncoming wavefront edge.
- Edge events: a wavefront edge vanishes.



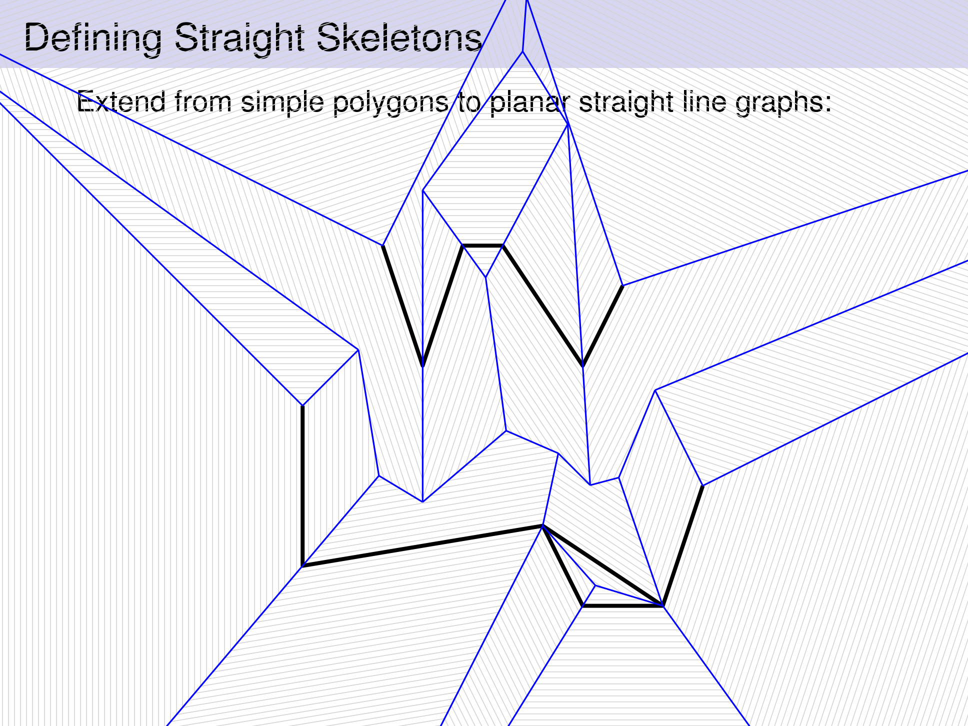
# Defining Straight Skeletons

Extend from simple polygons to planar straight line graphs:

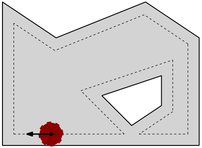


# Defining Straight Skeletons

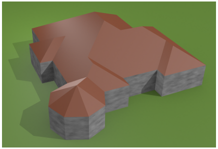
Extend from simple polygons to planar straight line graphs:



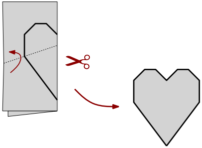
# Applications



Tool path generation



Roof construction

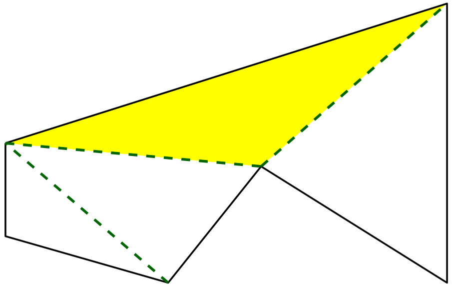


Cut-and-fold

and more . . .

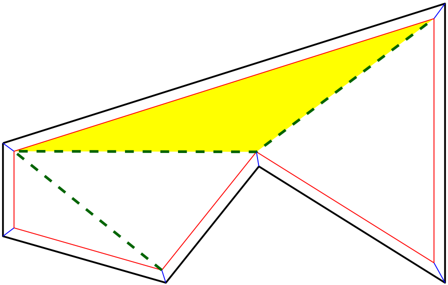
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



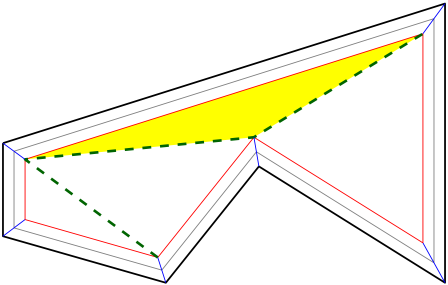
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



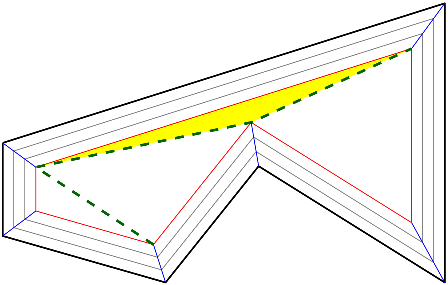
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



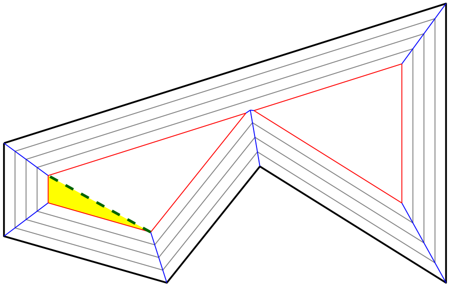
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



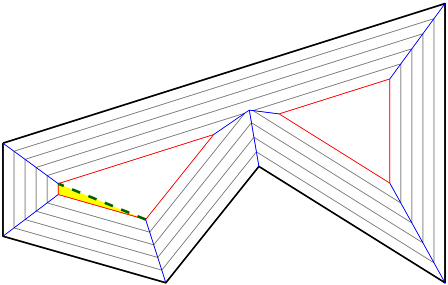
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



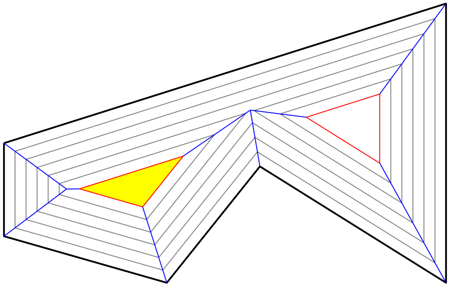
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles $\Rightarrow$ Priority Queue.



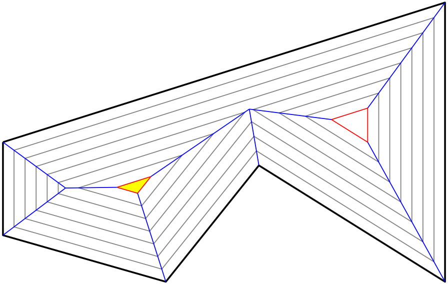
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



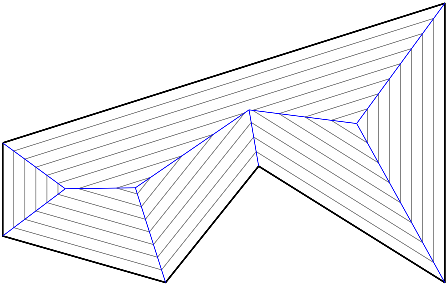
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



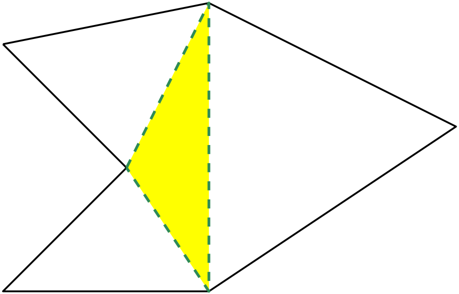
# Triangulation based Construction

- Keep a triangulation of the area not yet swept over by the wavefront [Aichholzer, Aurenhammer 1998].
- Edge and Split events are witnessed by collapsing triangles ⇒ Priority Queue.



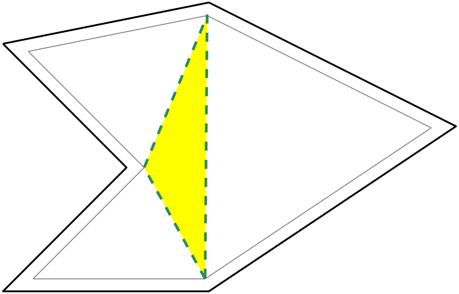
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



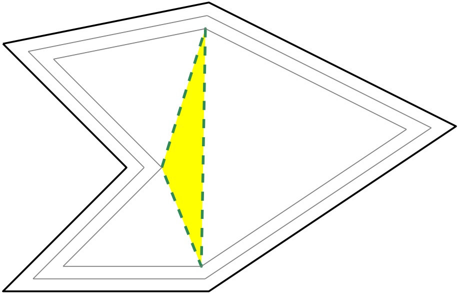
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



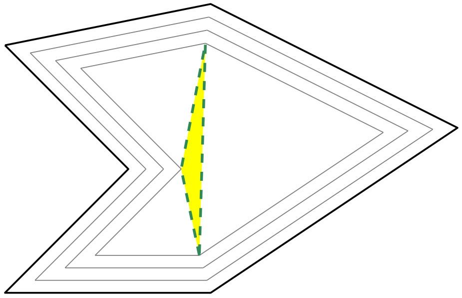
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



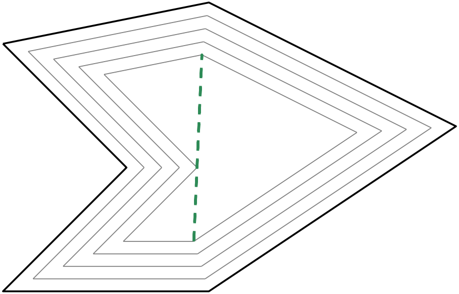
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



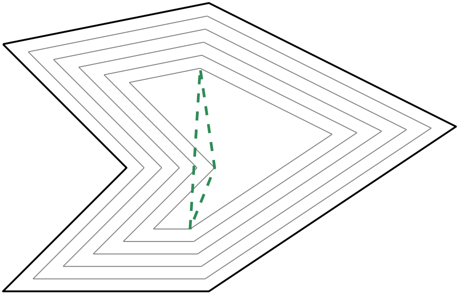
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



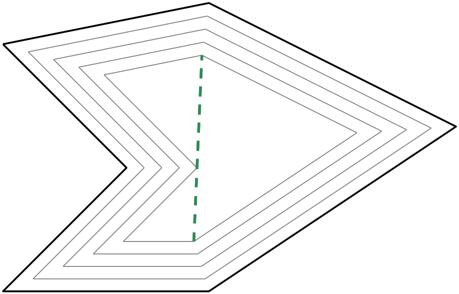
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



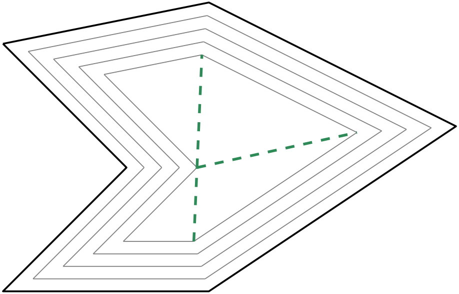
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



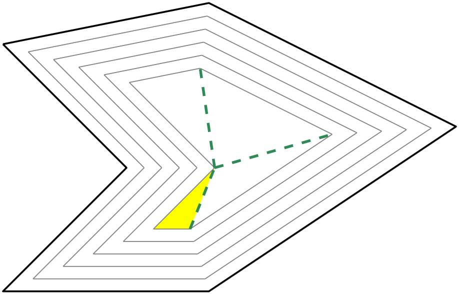
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



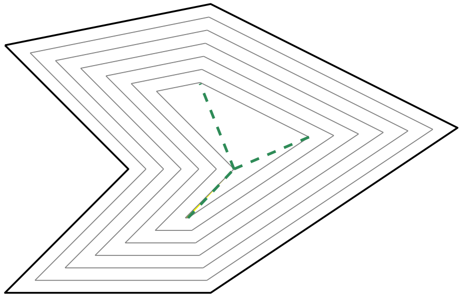
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



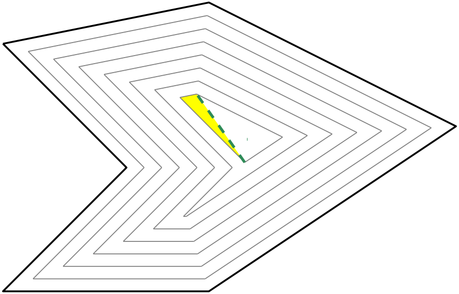
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



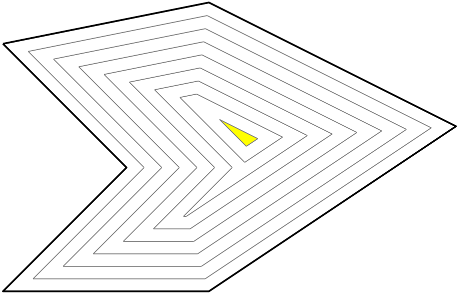
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



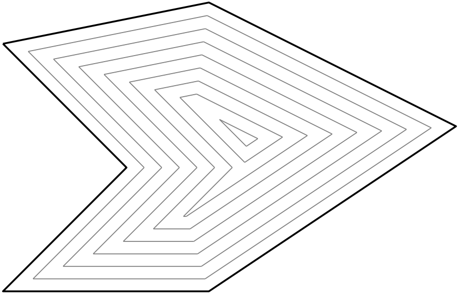
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



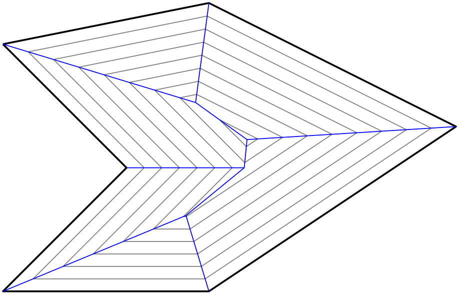
# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.



# Flip events

- Not all collapsing triangles result in a change of the wavefront.
- These cases need special handling nevertheless. They are *flip events*.

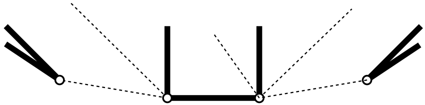


# Our contribution

- We have implemented this algorithm, filling in a few gaps in the algorithm, including issues that arise from not assuming general position.
- We have run extensive tests using this code, more on that in a bit.

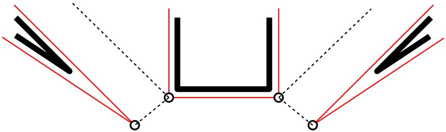
# Triangulating the input

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



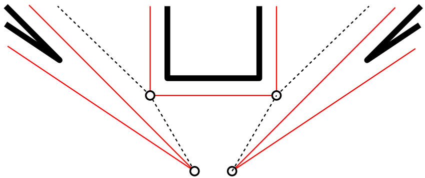
# Triangulating the input

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



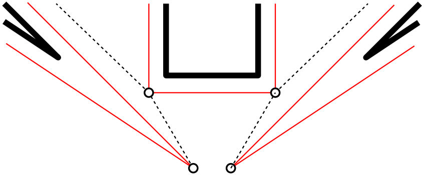
# Triangulating the input

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



# Triangulating the input

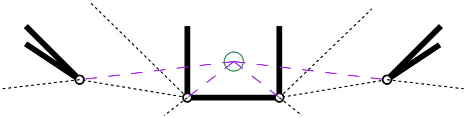
- Triangulate the convex hull.
- Unfortuantely the convex hull changes with time, and it matters.



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

# Triangulating the input

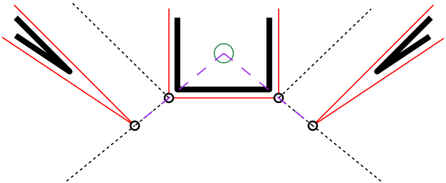
- Triangulate the convex hull.
- Unfortuantely the convex hull changes with time, and it matters.



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

# Triangulating the input

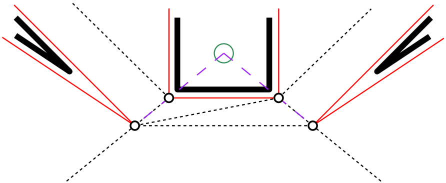
- Triangulate the convex hull.
- Unfortuantely the convex hull changes with time, and it matters.



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

# Triangulating the input

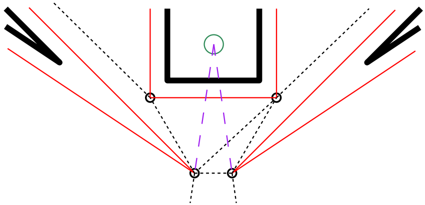
- Triangulate the convex hull.
- Unfortuantely the convex hull changes with time, and it matters.



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

# Triangulating the input

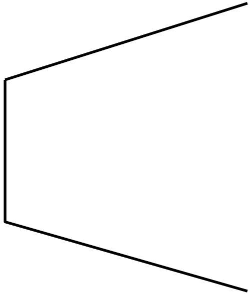
- Triangulate the convex hull.
- Unfortuantely the convex hull changes with time, and it matters.



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

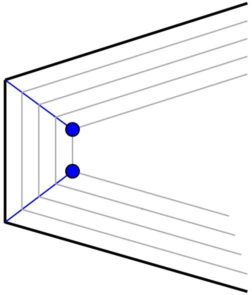
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.



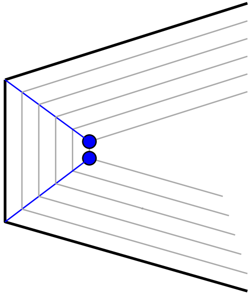
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.



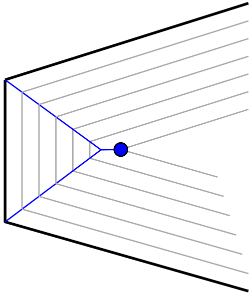
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.



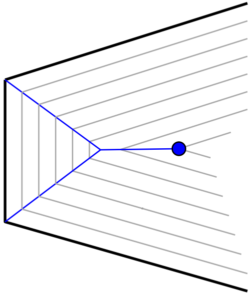
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.



# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.

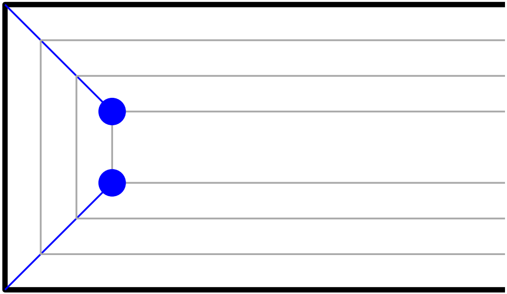


# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.
- But what if these edges are parallel?

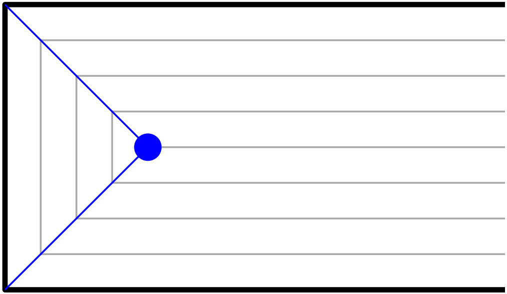
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.
- But what if these edges are parallel?



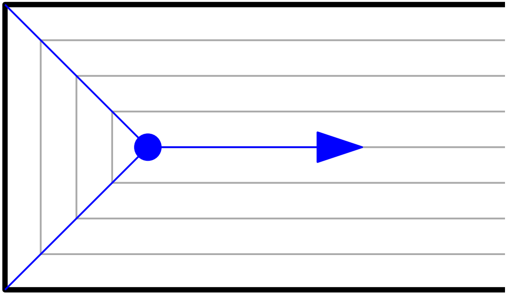
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.
- But what if these edges are parallel?



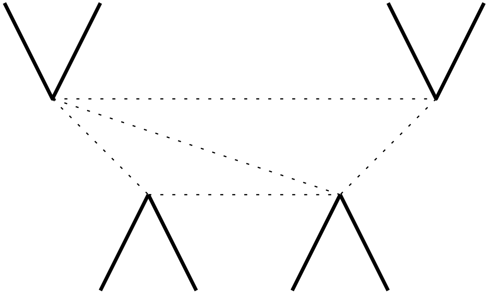
# Infinitely fast moving vertices

- During an edge event, two wavefront vertices collide and a new moving vertex gets launched.
- Its direction and speed is dictated by the incident wavefronts.
- But what if these edges are parallel?
- $\Rightarrow$ infinitely fast moving vertex.



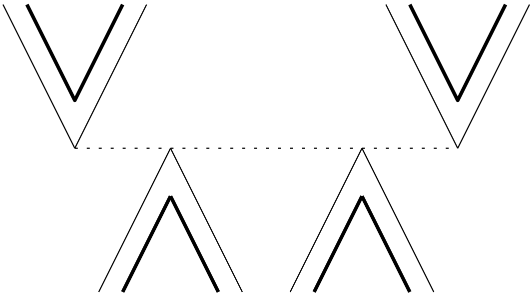
# Flip event loops

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



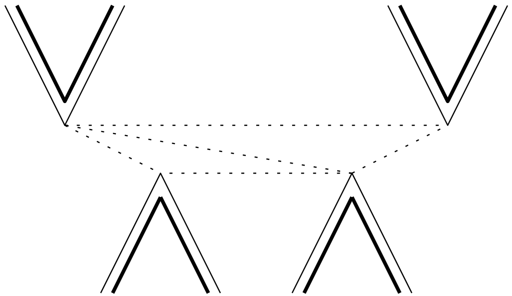
# Flip event loops

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



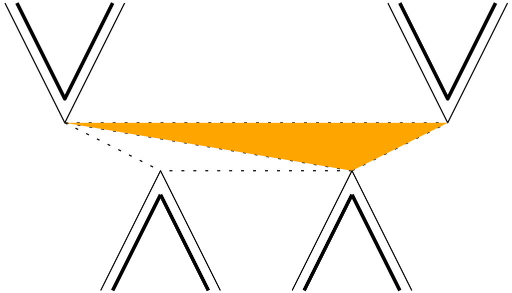
# Flip event loops

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



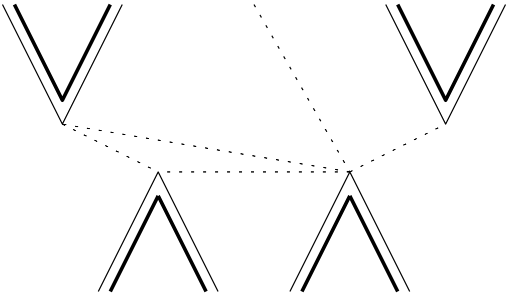
# Flip event loops

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



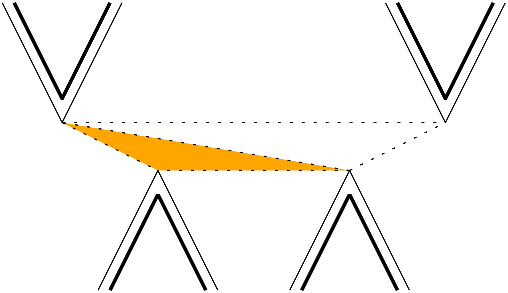
# Flip event loops

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



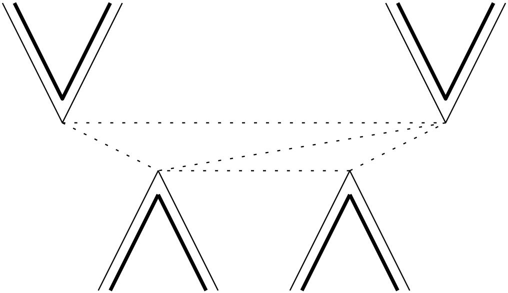
# Flip event loops

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



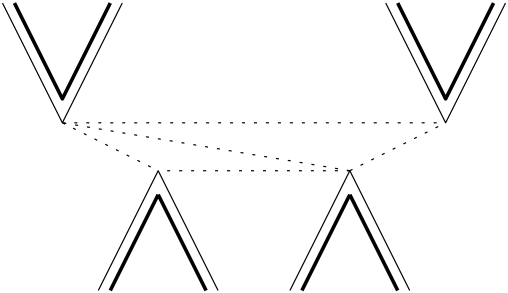
# Flip event loops

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



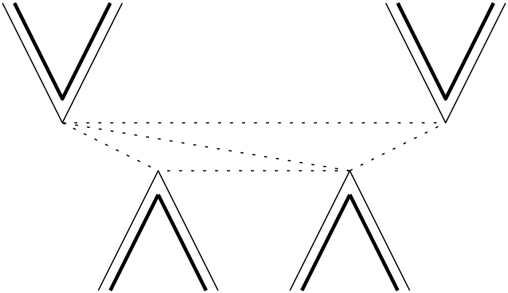
# Flip event loops

- 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 operatons but can also happen with exact arithmetic.

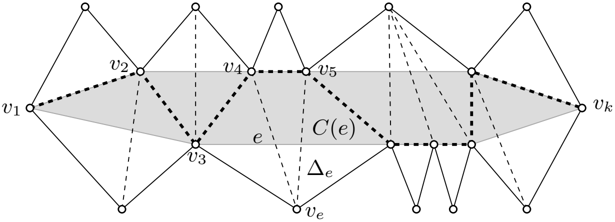
# Detecting flip event loops

- 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:

- $e_a$ [$e_b$] is the first [last] occurance of the duplicate event.
- Events between $e_a$ and $e_b$ happen at the same time.
- The set of triangles with events between $e_a$ and $e_b$ make up one or more edge-connected components.
- The component that contains the triangle of $e_a$ is a polygon $P$ which has collapsed to a straight line.
- Undo the events of the triangles in $P$, and retriangulate $P$ and its neighborhood.




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

# Number of flip events

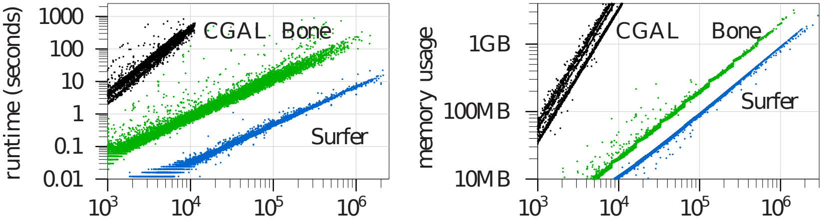
- Three points moving at constant speed become collinear at most twice.
- With $n$ vertices, there are $\binom{n}{3} \in \mathcal{O}(n^3)$ combinatorially different triangles.
- $\mathcal{O}(n^3)$ is the best known upper bound on the number of flip events!
- No input is known that results in more than quadratically many flip events.
- It turns out that for *practical data* the number of flip events is very linear.

# 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.