Computing

Straight Skeletons

by Means of

Kinetic Triangulations

Peter Palfrader

October 2013
Computing Straight Skeletons by Means of Kinetic Triangulations

1 Introduction
   Definition
   Applications

2 Triangulation-based Algorithm
   Basic Idea
   Flaws of the original Algorithm
   Experimental Results
• Problem: Given input graph, find the *straight skeleton*.
• Aichholzer, Alberts, Aurenhammer, Gärtner 1995.
• Problem: Given input graph, find the straight skeleton.
• input polygon $\mathcal{P}$ emanates wavefront $WF(\mathcal{P}, t)$. 
• input polygon $\mathcal{P}$ emanates wavefront $WF(\mathcal{P}, t)$.
• wavefront propagation — shrinking process.
• input polygon $\mathcal{P}$ emanates wavefront $WF(\mathcal{P}, t)$.
• wavefront propagation — shrinking process.
Straight Skeletons – Motivation

- input polygon $P$ emanates wavefront $WF(P, t)$.
- wavefront propagation — shrinking process.
**Straight Skeletons – Motivation**

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

• input polygon $\mathcal{P}$ emanates wavefront $WF(\mathcal{P}, t)$.
• wavefront propagation — shrinking process.
• straight skeleton $SK(\mathcal{P})$ is traces of wavefront vertices.
• Wavefront topology changes over time.
• Wavefront topology changes over time.
• Wavefront topology changes over time.
• Wavefront topology changes over time.
• *edge event*: an edge of $\mathcal{WF}(\mathcal{P}, t)$ vanishes.
Wavefront topology changes over time.

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

- Maintain a kinetic triangulation of the points of the plane not yet visited.
**Triangulation-based Algorithm**

- Maintain a kinetic triangulation of the points of the plane not yet visited.
**TRIANGULATION-BASED ALGORITHM**

- Maintain a kinetic triangulation of the points of the plane not yet visited.
TRIANGULATION-BASED ALGORITHM

• Maintain a kinetic triangulation of the points of the plane not yet visited.
Triangulation-based Algorithm

- Maintain a kinetic triangulation of the points of the plane not yet visited.
- Collapsing triangles witness edge and split events.
TRIANGULATION-BASED ALGORITHM

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

Maintain a kinetic triangulation of the points of the plane not yet visited.

Collapsing triangles witness edge and split events.

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

• Maintain a kinetic triangulation of the points of the plane not yet visited.
• Collapsing triangles witness edge and split events.
Maintain a kinetic triangulation of the points of the plane not yet visited.
Collapsing triangles witness edge and split events.

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

TRIANGULATION-BASED ALGORITHM
• Caveat: Not all collapses witness changes in the wavefront topology.
• Caveat: Not all collapses witness changes in the wavefront topology.
Caveat: Not all collapses witness changes in the wavefront topology.

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

Such collapses cannot be ignored.

Instead they need special processing: *flip events*.
• Caveat: Not all collapses witness changes in the wavefront topology.
• Such collapses cannot be ignored.
• Instead they need special processing: *flip events*. 
• 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.
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.
Without general position, this algorithm can end up in infinite loops.
Without general position, this algorithm can end up in infinite loops.
Without general position, this algorithm can end up in infinite 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.

![Diagram of flip-event loops](image-url)
Without general position, this algorithm can end up in infinite 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.
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

- $O(n^3)$ 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

<table>
<thead>
<tr>
<th></th>
<th>Theoretical Worst Case Runtime</th>
<th>Theoretical Worst Case Space</th>
<th>Practical Runtime</th>
<th>Practical Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>E&amp;E</td>
<td>(O(n^{17/11+\epsilon}))</td>
<td>(O(n^{17/11+\epsilon}))</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>CGAL</td>
<td>(O(n^2 \log n))</td>
<td>(O(n^2))</td>
<td>(O(n^2 \log n))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>Bone</td>
<td>(O(n^2 \log n))</td>
<td>(O(n))</td>
<td>(O(n \log n))</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Surfer</td>
<td>(O(n^3 \log n))</td>
<td>(O(n))</td>
<td>(O(n \log n))</td>
<td>(O(n))</td>
</tr>
</tbody>
</table>

---

1 Eppstein and Erickson, 1999
2 F. Cacciola, 2004
3 Huber and Held, 2010
4 this, based on Aichholzer and Aurenhammer, 1998
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 $O(n \log n)$ 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.
Thank you for your attention.
GALLERY: BORDERS OF AUSTRIA
GALLERY: PCB
GALLERY: POLYGON WITH HOLE
GALLERY: MORE HOLES
GALLERY: ALMOST POLYGON
GALLERY: STAR
GALLERY: SPIRALS
APPLICATIONS: GIS

image credit: Stefan Huber
MEDIAL AXIS VS. SK

VD-based MA

SK
**Alternate Computation**

![Diagram of a triangle with vectors and a graph showing function $f(t)$ over time with a note indicating $\Delta$ collapses at a specific time.]
INFINITELY FAST VERTICES
• Triangulate the convex hull.
• Unfortunately the convex hull changes with time, and it matters.
• Triangulate the convex hull.
• Unfortunately the convex hull changes with time, and it matters.
• Triangulate the convex hull.
• Unfortunately the convex hull changes with time, and it matters.
• 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?
• 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?
• 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?
• 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
$\Omega$ FOR NON-FLIP EVENTS

$\Omega(n)$ triangles

$\Omega(n)$ edge events

$S(P)$

$e_1, e_2, \ldots, e_k$
Pick, in order:

- non-flip event → reduces triangles
- longest edge to flip → reduces longest edge (count or length)
Affected Triangles, Max

In edge events

In split events
Affected Triangles, Avg
TIME SPENT, PHASES

runtime (percentage of total)

pre-processing  triangulation  kinetic triangulation  initial schedule  propagation process  post-processing
MPFR

slowdown

blowup