Computing Straight Skeletons of Planar Straight-Line Graphs Based on Motorcycle Graphs

CCCG2010, Winnipeg, Canada

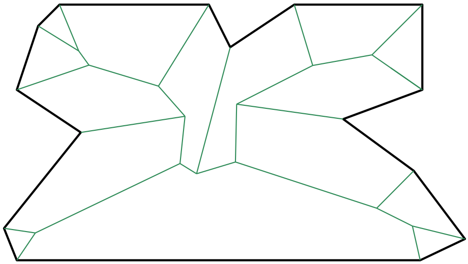
Stefan Huber   Martin Held

Universität Salzburg, Austria

August 11, 2010

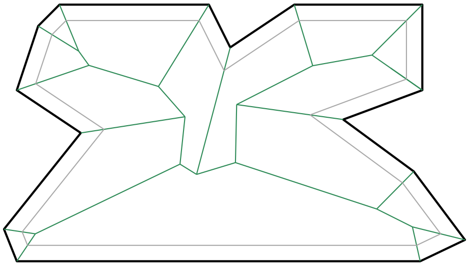
Straight skeleton of a simple polygon

- Aichholzer et alii [1995]: straight skeleton of simple polygons.
- Similar to Voronoi diagram, but consists only of straight-line segments.



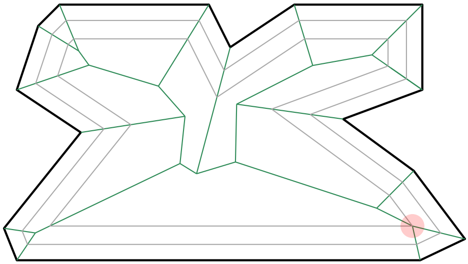
Straight skeleton of a simple polygon

- Aichholzer et alii [1995]: straight skeleton of simple polygons.
- Similar to Voronoi diagram, but consists only of straight-line segments.
- Self-parallel wavefront propagation process.



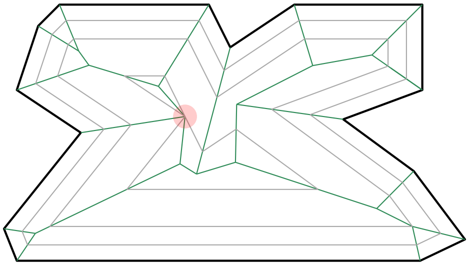
Straight skeleton of a simple polygon

- Aichholzer et alii [1995]: straight skeleton of simple polygons.
- Similar to Voronoi diagram, but consists only of straight-line segments.
- Self-parallel wavefront propagation process.
- Topological changes:
    - edge events



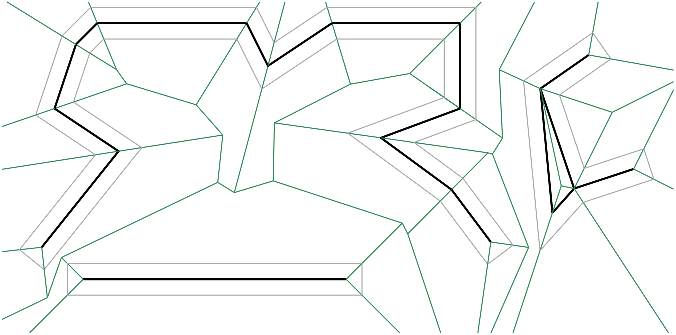
Straight skeleton of a simple polygon

- Aichholzer et alii [1995]: straight skeleton of simple polygons.
- Similar to Voronoi diagram, but consists only of straight-line segments.
- Self-parallel wavefront propagation process.
- Topological changes:
  - edge events
  - split events



Straight skeleton of planar straight-line graphs

- Aichholzer and Aurenhammer [1998]: generalization to planar straight-line graphs.



Status quo

Consider a **planar straight-line graph** with n vertices as input. No vertex shall be isolated.

Algorithms with sub-quadratic runtime:

- Eppstein & Erickson, $O(n^{17/11+\epsilon})$ runtime. Very complex, no implementation known.

Algorithms capable for implementation:

- Aichholzer & Aurenhammer, $O(n^3 \log n)$ runtime. Easy to implement. In general, seems to be fast for many input datasets.

Status quo

Consider a **planar straight-line graph** with n vertices as input. No vertex shall be isolated.

Algorithms with sub-quadratic runtime:

- Eppstein & Erickson, $O(n^{17/11+\epsilon})$ runtime. Very complex, no implementation known.

Algorithms capable for implementation:

- Aichholzer & Aurenhammer, $O(n^3 \log n)$ runtime. Easy to implement. In general, seems to be fast for many input datasets.

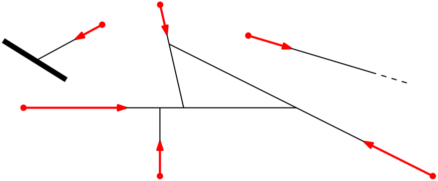
Our contribution

Worst-case runtime of $O(n^2 \log n)$. Easy to implement. Experiments show an actual $O(n \log n)$ runtime in practice.

Motorcycle graph

A **motorcycle** is a point moving constantly on a straight line and leaves a **trace** behind. A motorcycle stops (**crashes**) when reaching the trace of another motorcycle. The arrangement of the traces is called motorcycle graph.

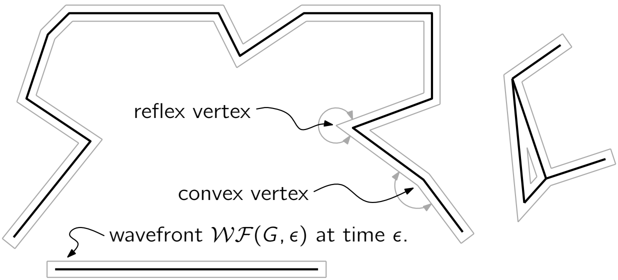
Addition: motorcycles may also crash against solid straight-line segments (**walls**).



- Introduced by Eppstein & Erickson as the essential sub problem of straight skeletons.
- Used by Cheng & Vigneron to compute straight skeletons of "non-degenerated" polygons with holes in $O(n\sqrt{n}\log^2 n)$ time.

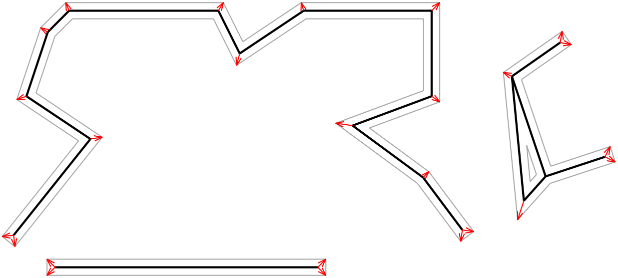
Motorcycle graph on the PSLG

- Input graph denoted by G.
- Consider some time ϵ such that no event happened so far.



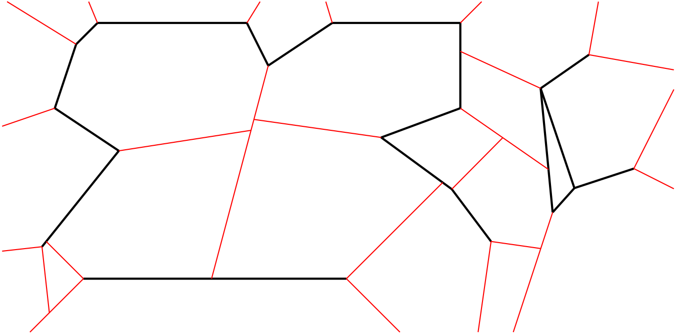
Motorcycle graph on the PSLG

- Input graph denoted by G.
- Consider some time ϵ such that no event happened so far.
- For a vertex v of the wavefront, $v(t)$ denotes position of v at time t.
- For every reflex vertex v of $\mathcal{W}(G, \epsilon)$ we define a motorcycle starting at $v(0)$ and with speed vector ${}^{(v(\epsilon)-v(0))}/_{\epsilon}$.
- Further, consider the edges of G as walls.



Motorcycle graph on the PSLG

We denote the resulting motorcycle graph by $\mathcal{M}(G)$.

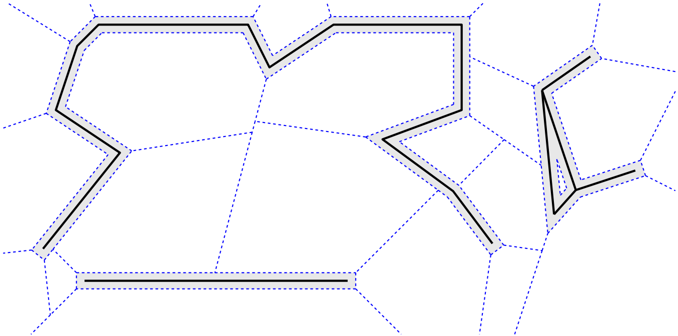


Assumption

We adopt the assumption of Cheng & Vigneron:
No two motorcycles crash simultaneously at a common location.

Computing the straight skeleton

- We denote by $\mathcal{M}(G,t)$ those parts of $\mathcal{M}(G)$ which have not been swept by the wavefront until time t.
- Let $\mathcal{W}^*(G,t)$ denote the overlay of $\mathcal{W}(G,t)$ and $\mathcal{M}(G,t)$.



- Simulating the propagation of $\mathcal{W}(G,t)$ in the straight-forward manner is inefficient. (Split events.)
- We simulate the propagation of $\mathcal{W}^*(G,t)$ instead.

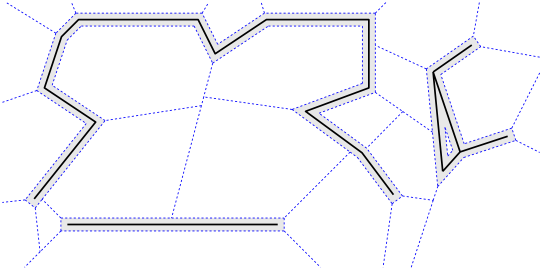
Basic properties

Theorem (Cheng & Vigneron)

Reflex straight skeleton arcs are shorter than the corresponding motorcycle traces.

Corollary

Split events happen within the corresponding motorcycle traces and consequently within the extended wavefront $\mathcal{W}^*(G, t)$.



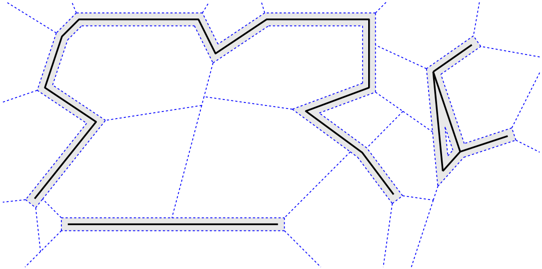
Basic properties

Lemma

For any $t \geq 0$ the set $\mathbb{R}^2 \setminus \bigcup_{t' \in [0,t]} \mathcal{W}^*(G, t')$ consists of open convex faces.

Corollary

During the propagation of $\mathcal{W}^*(G, t)$ only neighboring vertices can meet.



Basic algorithm

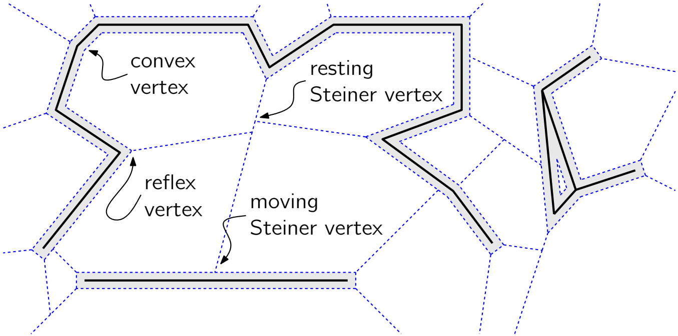
Event: a topological change of $\mathcal{W}^*(G,t)$, i.e. an edge of $\mathcal{W}^*(G,t)$ collapsed to zero length.

Algorithm

1. Compute the initial extended wavefront $\mathcal{W}^*(G,0)$.
2. Compute for every edge of $\mathcal{W}^*(G,0)$ the collapsing time t. If $t \in (0,\infty)$ then insert the corresponding event into a **priority queue** Q.
3. Fetch the earliest event of Q. Process the event, i.e. maintain $\mathcal{W}^*(G,t)$ and possibly insert new events into Q. Repeat until Q is empty.

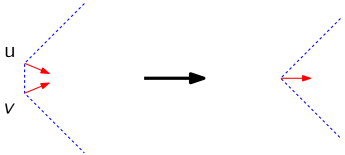
Algorithmic details: types of vertices

We distinguish the following types of vertices:



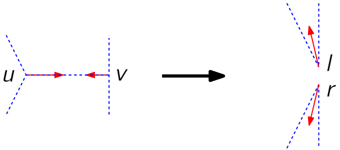
Algorithmic details: Types of events

- **(Classical) edge event:** two convex vertices meet



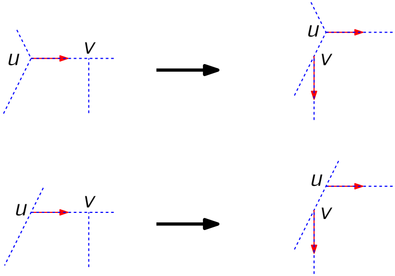
Algorithmic details: Types of events

- **(Classical) edge event:** two convex vertices meet
- **(Classical) split event:** a reflex and a moving Steiner vertex meet



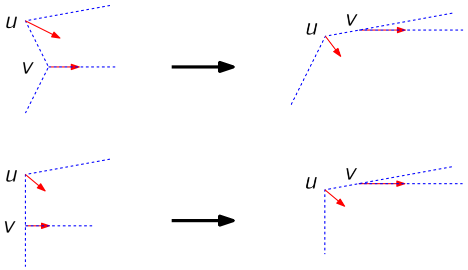
Algorithmic details: Types of events

- **(Classical) edge event:** two convex vertices meet
- **(Classical) split event:** a reflex and a moving Steiner vertex meet
- **Start event:** a reflex or a moving Steiner vertex u meets a resting Steiner vertex v. The vertex v becomes a moving Steiner vertex.



Algorithmic details: Types of events

- **(Classical) edge event:** two convex vertices meet
- **(Classical) split event:** a reflex and a moving Steiner vertex meet
- **Start event:** a reflex or a moving Steiner vertex u meets a resting Steiner vertex v. The vertex v becomes a moving Steiner vertex.
- **Switch event:** a convex vertex u meets a reflex or a moving Steiner vertex v. The vertex u migrates to a neighboring convex face. If v was a reflex vertex then it becomes a moving Seiner vertex.



Runtime complexity

Assume the motorcycle graph of $\mathcal{M}(G)$ is given.

- Every vertex in $\mathcal{W}^*(G, t)$ has degree at most three. Hence every event is processed in $O(\log n)$ time.
- Edge, split and start events occur in total $\Theta(n)$ times and hence consume $O(n \log n)$ time.
- A convex vertex does not meet a moving Steiner point twice. Hence, the number k of switch events is in $O(n^2)$.

Lemma

If $\mathcal{M}(G)$ is given our algorithm takes $O((n+k)\log n)$ time, where k is the number of switch events, with $k \in O(n^2)$.

- For practical input it seams unlikely that more than $O(n)$ switch events occur, as confirmed by experiments.
- A worst-case example can be constructed.

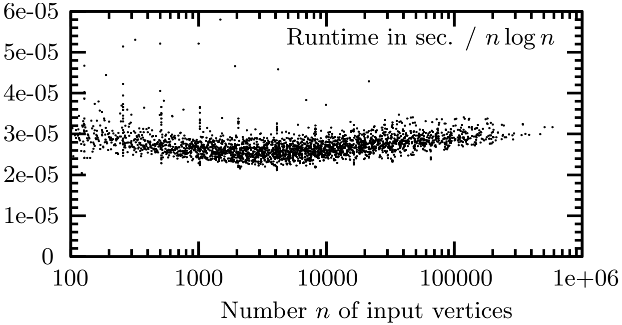
Runtime complexity

Computing the motorcycle graph $\mathcal{M}(G)$:

- Priority queue enhanced straight-forward algorithm takes $O(n^2 \log n)$ time.
- Sub-quadratic algorithms are given by Eppstein & Erickson and Cheng & Vigneron.
- Our implementation Moca has an average runtime of $O(n \log n)$.

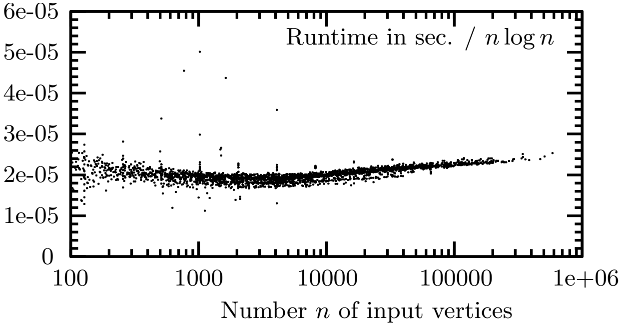
Experimental results

- Implemented in C++. $\mathcal{M}(G)$ is computed by our code Moca.
- 3 100 datasets of different flavors.
- Total runtime, including the computation of $\mathcal{M}(G)$:



- Implemented in C++. $\mathcal{M}(G)$ is computed by our code Moca.
- 3 100 datasets of different flavors.

- Runtime for the computation of $\mathcal{S}(G)$ only, excluding the computation of $\mathcal{M}(G)$:



Conclusion

Summary:

- Handles PSLG as input.
- Easy to implement, with a worst-case runtime of $O(n^2 \log n)$ instead of $O(n^3 \log n)$.
- Promising experimental results showing an $O(n \log n)$ runtime for practical input.

Future work:

- Getting rid of the assumption of Cheng & Vigneron: Implementation done, missing theoretical work in progress.

Appendix

Worst-case runtime complexity