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

Stefan Huber† Martin Held†

Abstract

We present a simple algorithm for computing straight skeletons of planar straight-line graphs. We exploit the relation between motorcycle graphs and straight skeletons, and introduce a wavefront-propagation algorithm that circumvents the expensive search for the next split event. Our algorithm maintains the simplicity of the triangulation-based algorithm by Aichholzer and Aurenhammer but has a better worst-case complexity of $O(n^2 \log n)$. Preliminary experiments with our implementation demonstrate that an actual runtime of $O(n \log n)$ can be expected in practice.

1 Introduction

Straight skeletons of simple polygons were introduced by Aichholzer et al. [1] and later generalized to planar straight-line graphs by Aichholzer and Aurenhammer [2]. For planar straight-line graphs G with n vertices two algorithms are known: an $O(n^3 \log n)$ algorithm by Aichholzer and Aurenhammer [2] and an $O(n^{17/11+\epsilon})$ algorithm by Eppstein and Erickson [4]. The former algorithm uses a triangulation of G to perform a wavefront propagation; it is widely believed yet unproven that its worst-case complexity is in $O(n^2 \log n)$, cf. [6]. The latter algorithm exploits dynamic data structures for fast nearest-neighbor queries and is quite complex when considering all its sub-algorithms. In particular, no implementation is known that matches the theoretical complexity. Cheng and Vigneron [3] introduced a randomized algorithm for simple polygons with holes. They exploit the relation between motorcycle graphs and straight skeletons to obtain an expected runtime of $O(n\sqrt{n} \log^2 n)$. Again, no implementation of their algorithm is known.

Let us consider a planar straight-line graph G with n vertices, none of them being isolated. Vertices of degree one are called terminals. According to [2], the definition of the straight skeleton $\mathcal{S}(G)$ is based on a wavefront-propagation process. Roughly speaking, every edge e of G sends two wavefronts out, which are parallel to e and have unit speed. At terminals of G an additional wavefront orthogonal to the single incident edge is emitted. In Fig. 1 we illustrated the wavefront of an input graph at three different points in time. The wavefront $\mathcal{W}(G,t)$ of G at some time t can be interpreted as a 2-regular kinetic straight-line graph, where the vertices of $\mathcal{W}(G,t)$ move along bisectors of straight-line edges of G (except for the vertices originating from the terminals of G). During the propagation of $\mathcal{W}(G,t)$ topological changes occur: an edge may collapse (edge event) or an edge may be split by a vertex (split event). The straight-line segments traced out by the vertices of $\mathcal{W}(G,t)$ form the so-called straight skeleton $\mathcal{S}(G)$ of G, see Fig. 1.

Aichholzer and Aurenhammer [2] gave a versatile interpretation of $\mathcal{S}(G)$ by considering a fixed-slope terrain in $\mathbb{R}^3$ using the following construction. They embed G and $\mathcal{S}(G)$ in the plane $\mathbb{R}^2 \times \{0\}$. Now assume that the propagating wavefronts $\mathcal{W}(G,t)$ of G are moving upwards at unit speed. Then the wavefronts contour a fixed-slope terrain $\mathcal{T}(G) \subset \mathbb{R}^3$ of the form $\bigcup_{t \geq 0} \mathcal{W}(G,t) \times \{t\}$. The wavefront at some time t can be interpreted as the isoline of $\mathcal{T}(G)$ at height t. The straight skeleton $\mathcal{S}(G)$ is given by the projection of the valleys and ridges of $\mathcal{T}(G)$ onto the plane.

We call an edge e of $\mathcal{S}(G)$ reflex (convex, resp.) if the corresponding edge $\hat{e}$ in $\mathcal{T}(G)$ is a valley (ridge, resp.). Further, we call a vertex v of $\mathcal{W}(G,t)$ reflex (convex, resp.) if the angle between the two incident edges on the side where v propagates to is $\geq 180^o$ ($< 180^o$, resp.). Hence, reflex edges of $\mathcal{S}(G)$ are traced out by reflex vertices of $\mathcal{W}(G,t)$.

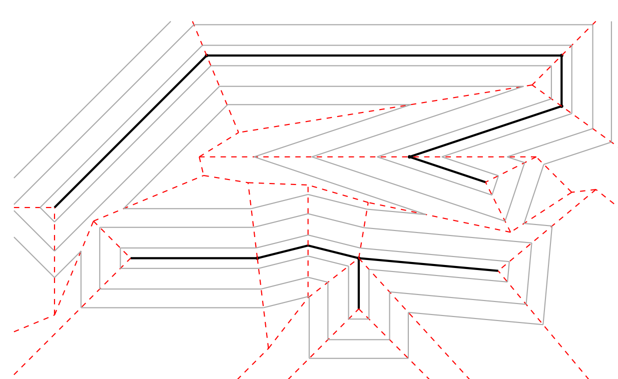

Figure 1: The straight skeleton (dashed) of the input graph (bold) and three wavefronts (grey).

*Work supported by Austrian FWF Grant L367-N15.

†Universität Salzburg, FB Computerwissenschaften, A-5020 Salzburg, Austria, {shuber,held}@cosy.sbg.ac.at

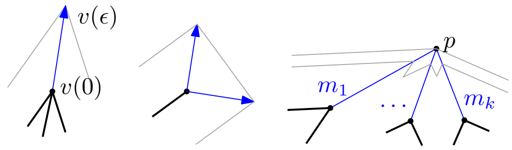

Figure 2: Starting a new motorcycle. (a) a motorcycle is launched because v is reflex. (b) at terminals of G two motorcycles are launched. (c) forbidden case: motorcycles $m_1, \dots, m_k$ crash simultaneously at p.

In a previous paper [6] we demonstrated how to employ Steiner points in order to eliminate all flip events from the triangulation-based algorithm of [2], and sketched an algorithm for computing straight skeletons of simple polygons by means of motorcycle graphs. In this paper we discuss this algorithm in detail, extend it to planar straight-line graphs and report on experiments. The basic idea is to simulate the wavefront propagation on the motorcycle graph in order to make the handling of a topological event computationally cheap.

2 Motorcycle graph of a PSLG

In order to extend the approach of our previous work [6] to planar straight-line graphs we extend a result of Cheng and Vigneron [3], see Thm. 1. Consider a set of points in the plane, called "motorcycles", that drive along straight-line rays according to given speed vectors. Further consider a set of straight-line segments, called "walls". Every motorcycle leaves a trace behind it and stops driving — it "crashes" — when reaching the trace of another motorcycle or a wall. The arrangement of these traces is called motorcycle graph, cf. [5].

We denote by $v(t)$ the position of the vertex v of $\mathcal{W}(G, t)$. Let us consider $\mathcal{W}(G, \epsilon)$ for a sufficiently small $\epsilon > 0$ such that no topological event occurred yet in the wavefront propagation. Then $\mathcal{W}(G, \epsilon)$ is a 2-regular planar straight-line graph. For every reflex vertex v of $\mathcal{W}(G, \epsilon)$ we define a motorcycle with start point $v(0)$ and speed vector $\frac{1}{\epsilon}(v(\epsilon) - v(0))$, see Fig. 2. In particular, at every terminal of G two motorcycles are launched. Next, we consider the edges of G as walls. We denote the motorcycle graph resulting from this setup by $\mathcal{M}(G)$.

For the sake of simplicity we adopt the assumption of Cheng and Vigneron [3]: we assume that no two or more motorcycles crash simultaneously at some point p. Hence, the case in Fig. 2 (c) is excluded. In particular, this means that no two or more reflex vertices of $\mathcal{W}(G, t)$ meet simultaneously at some point p.

Theorem 1 *Every reflex edge in $\mathcal{S}(G)$ is covered by a motorcycle trace in $\mathcal{M}(G)$.*

The following proof is given in [3] and has been slightly adapted to our purposes.

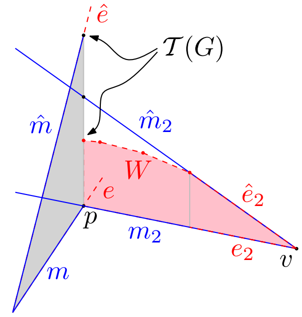

Figure 3: The terrain $\mathcal{T}(G)$ has two different heights at position p, given by $\hat{e}$ and W, which is a contradiction.

Proof. As in [3], the problem is lifted to $\mathbb{R}^3$ by considering $\mathcal{T}(G)$. We denote by $\hat{e}$ the edge of $\mathcal{T}(G)$ which corresponds to the edge e in $S(G)$. Analogously, we denote by $\hat{m}$ the tilted version of the motorcycle m, where the slope is the reciprocal of the speed of m, see Fig. 3.

We first note that every reflex edge of $\mathcal{S}(G)$ is incident to a (reflex) vertex of $\mathcal{W}(G, 0)$. (Topological events in the wavefront propagation do not lead to new reflex edges.) This means that every reflex edge of $\mathcal{S}(G)$ corresponds to a unique motorcycle trace in $\mathcal{M}(G)$. Hence every valley of $\mathcal{T}(G)$ corresponds to a unique motorcycle trace, except for the valleys lying on the plane, which correspond to the input graph G.

Assume that there is a reflex edge e of $\mathcal{S}(G)$ which is strictly shorter than the trace of the corresponding motorcycle m. Within all such candidates we consider an edge e where the upper endpoint of $\hat{e}$ has minimum height. Obviously, the upper endpoint is part of $\mathcal{T}(G)$. Since m is shorter than e it follows that m did not crash against a wall (an edge of G) but against another motorcycle m_2. We denote by $p \in \mathbb{R}^2$ the crash point, by e_2 the edge of $\mathcal{S}(G)$ corresponding to m_2 and by v the start point of m_2. Further let t^* denote the height of the lower endpoint of $\hat{e}$.

Let us consider the vertical slab W having the base line $[v, p]$ and being bounded above by $\mathcal{T}(G)$, see Fig. 3. We observe that $W \cap T(G)$ is convex: consider the corresponding vertices in the order as they appear on m_2. First, we consider the higher endpoint of $\hat{e}_2$. If this vertex would be reflex we would have discovered a valley of $\mathcal{T}(G)$. Since the corresponding point is lower than t^* there has to be a corresponding motorcycle m_3 such that $\hat{m}_3$ reaches this point. But then m_2 and m_3 would have crashed simultaneously, which is excluded by the assumption of Cheng and Vigneron. Every further vertex of $W \cap T(G)$ is strictly below $\hat{m}_2$ by induction and the following argument. Assume that such a vertex would be reflex. Since motorcycles crash at walls (edges of G) we can not have reached height zero. Hence there would again be a motorcycle m_3 which reaches the corresponding position strictly before m_2 and hence m_2 would have been crashed against m_3 but never reached p.

Finally, $W \cap \mathcal{T}(G)$ is at position p not above $\hat{m}_2$. But on the other hand the height of $T(G)$ is given at p by the lower endpoint of $\hat{e}$ which is a contradiction. □

3 Computing the straight skeleton

First, we add $\mathcal{M}(G)$ to the wavefront by the following construction. Consider for a $t \geq 0$ those parts $\mathcal{M}(G,t)$ of $\mathcal{M}(G)$ which have not yet been swept by the wavefront $\mathcal{W}(G,t)$ and insert $\mathcal{M}(G,t)$ into $\mathcal{W}(G,t)$ by splitting the edges of $\mathcal{W}(G,t)$ at the intersection points. Those intersection points are called moving Steiner vertices. Each vertex of $\mathcal{M}(G)$ not lying on $\mathcal{W}(G,t)$ is due to a crash of a motorcycle into the trace of another motorcycle and will be called resting Steiner vertices. The resulting graph will be denoted by $\mathcal{W}^*(G,t)$, see Fig. 4. Again $\mathcal{W}^*(G,t)$ can be interpreted as a kinetic planar straight-line graph.

Lemma 2 *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.*

This is easy to see since reflex angles at reflex vertices of $\mathcal{W}(G,t)$ are split by (parts of) motorcycle traces accordingly. In particular, for $t = 0$, the lemma implies that $G + \mathcal{M}(G)$ induces a tessellation of the plane into (possibly unbounded) convex faces. A consequence of the lemma is that during the propagation of $\mathcal{W}^*(G,t)$ only adjacent vertices can meet.

Our straight skeleton algorithm simply simulates the propagation of $\mathcal{W}^*(G,t)$. While simulating the original wavefront $\mathcal{W}(G,t)$ leads to the problem of finding the next split event (a reflex vertex meets a wavefront edge) we circumvent this problem due to Lemma 2: every topological change is indicated by the collision of two neighboring vertices of $\mathcal{W}^*(G,t)$. Theorem 1 guarantees that a split event occurs within the corresponding motorcycle trace and hence reflex vertices do not move beyond them.

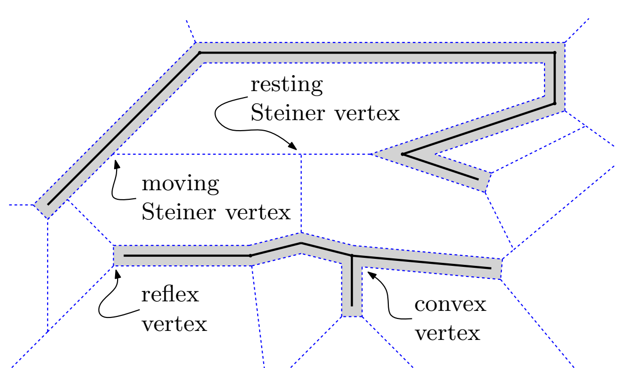

Figure 4: The graph $\mathcal{W}^*(G,t)$ (dotted) is the embedding of $\mathcal{M}(G,t)$ into $\mathcal{W}(G,t)$ for the input graph (bold). The shaded area has been already swept by $\mathcal{W}^*(G,t)$.

In order to compute $\mathcal{S}(G)$, we put every edge e of $\mathcal{W}^*(G,t)$ into a priority queue Q where the priority is given by the collapsing time of e. We fetch the next event in Q, apply the corresponding topological change to $\mathcal{W}^*(G,t)$ and repeat until Q gets empty. We consider the following event classes[1]:

(Classical) edge event Two convex vertices u and v meet. We add the convex straight skeleton arcs traced out by u and v. Then we merge u and v to a new convex vertex. As a special case we check whether a whole triangle collapsed due to u and v.

(Classical) split event A reflex vertex u meets a moving Steiner vertex v and both are driving against each other. First, we add a reflex straight skeleton arc which has been traced out by u. Then we consider the left side of the edge $e = (u,v)$. If this side collapsed we add corresponding straight skeleton arcs. Otherwise a new convex vertex emerges, which is connected to the vertices adjacent to u and v lying left of e. Likewise for the right side of e.

Start event A reflex vertex or a moving Steiner vertex u meets a resting Steiner vertex v. So v becomes a moving Steiner vertex and one of the incident edges of u (but not (u,v)) is split by v.

Switch event A convex vertex u meets a moving Steiner vertex or a reflex vertex v. The convex vertex u is migrating from one convex face to a neighboring one by jumping over v. If v was a reflex vertex then it becomes a moving Steiner vertex.

Remaining events If two moving Steiner vertices meet we can simply remove the corresponding edge. All other events (e.g. a convex vertex meets a resting Steiner vertex) are guaranteed not to occur.

For each event we have to update Q for those edges where the collapsing times change. Note that only $O(1)$ edges are changed per event. Therefore a single event is handled in $O(\log n)$ time. Edge, split and start events occur in total $\Theta(n)$ times. Since a single convex vertex does not meet a moving Steiner vertex twice the number of switch events is in $O(n^2)$. The construction of the initial wavefront $\mathcal{W}^*(G,t)$ can be done easily in $O(n \log n)$ time.

Lemma 3 *If $\mathcal{M}(G)$ is given then our algorithm takes $O((n+k)\log n)$ time, where k is the number of switch events, with $k \in O(n^2)$.*

[1] For technical reasons we have a further vertex type "multi convex vertex" in our implementation. It is used when a moving Steiner vertex and a convex vertex move identically. We do not further discuss this technical detail here.

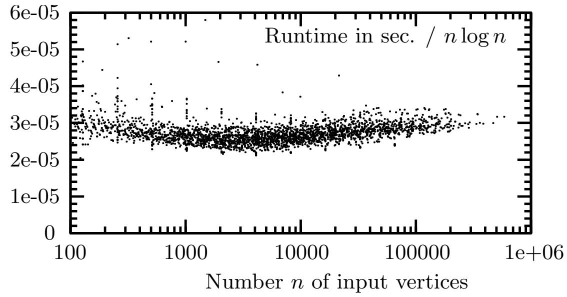

Figure 5: A point depicts the runtime on one dataset. The runtime is given in seconds and is scaled by $n \log n$.

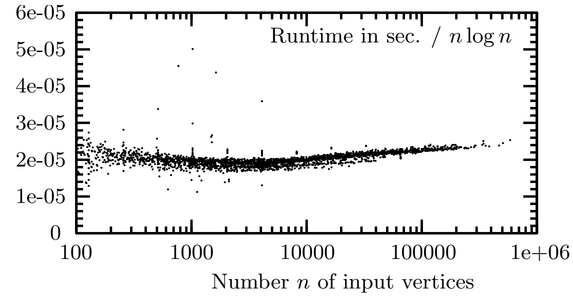

Figure 6: The same plot as in Fig. 5 but without the runtime for the motorcycle graph.

For practical applications it seems unlikely that more than $O(n)$ switch events occur and hence an actual runtime of $O(n \log n)$ may be expected, as confirmed by experiments; see below. (However, a worst-case example for the number of switch-events can be constructed.)

Sub-quadratic algorithms for the computation of $\mathcal{M}(G)$ are given in [4, 3]. Besides, it is well known that $\mathcal{M}(G)$ can be computed in $O(n^2 \log n)$ time by a priority-queue enhanced brute-force algorithm, cf. [3].

4 Experimental results

We have implemented our algorithm in C++ using ordinary double-precision floating-point arithmetic and the STL for standard data structures. The motorcycle graph is computed by our code Moca, which has an average runtime[2] of $O(n \log n)$.

The following runtime experiments have been done on a 32-bit Debian Linux machine with a 2.66 GHz Core Duo processor. We used the C function `getrusage()` to obtain the user time consumption. Our implementation is still under development. However, the current code is already mature enough to allow a glimpse at the runtime for about 3 100 datasets.

In Fig. 5 we plotted the runtime in seconds of our implementation (including the computation of the motorcycle graph). For a better illustration we scaled the values by a factor $n \log n$. It turns out that our implementation takes about $30\, n \log n\, \mu s$ on almost all datasets. In Fig. 6 we excluded the time taken by the computation of the motorcycle graph. About $20\, n \log n\, \mu s$ are used to compute $\mathcal{S}(G)$ if $\mathcal{M}(G)$ is already known. In both figures only datasets with at least 100 vertices have been plotted since the runtime is hardly measurable for smaller datasets. If the runtime for a single dataset was less than 0.1 seconds our code was launched multiple times and the average runtime was taken.

[2] While experiments in [5] already showed this runtime behavior we were able to improve the corresponding stochastic analysis. A full version of that paper is currently under review for publication.

5 Conclusion

In this paper we discuss a simple algorithm for the computation of the straight skeleton of a planar straight-line graph. As the algorithm by Aichholzer and Aurenhammer [2] our algorithm is suitable for implementation but its worst-case runtime is $O(n^2 \log n)$ instead of $O(n^3 \log n)$ for an n-vertex PSLG. Experiments with our C++ implementation on a few thousand datasets demonstrate a runtime of about $O(n \log n)$ in practice.

References

[1] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. Straight Skeletons of Simple Polygons. In *Proc. 4th Internat. Symp. of LIESMARS*, pages 114–124, Wuhan, P.R. China, 1995.

[2] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In A. Samoilenko, editor, *Voronoi's Impact on Modern Science, Book 2*, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiev, Ukraine, 1998.

[3] S.-W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. *Algorithmica*, 47(2):159–182, 2007.

[4] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete Comput. Geom.*, 22(4):569–592, 1999.

[5] S. Huber and M. Held. A Practice-Minded Approach to Computing Motorcycle Graphs. In *Proc. 25th Europ. Workshop Comput. Geom.*, pages 305–308, Brussels, Belgium, Mar 2009.

[6] S. Huber and M. Held. Straight Skeletons and their Relation to Triangulations. In *Proc. 26th Europ. Workshop Comput. Geom.*, pages 189–192, Dortmund, Germany, Mar 2010.