On Merging Straight Skeletons

Franz Aurenhammer[1] and Michael Steinkogler[2]

1 **Institute for Theoretical Computer Science, University of Technology, Graz, Austria**
auren@igi.tugraz.at

2 **Institute for Theoretical Computer Science, University of Technology, Graz, Austria**
steinkogler@igi.tugraz.at

Abstract

The search for efficient algorithms to compute the straight skeleton of a simple polygon has resulted in a variety of algorithms. We present a new approach that applies the divide-and-conquer paradigm with the divide step based on the motorcycle graph. A practical randomized algorithm is obtained that derives the straight skeleton from the motorcycle graph, with an expected running time of $\mathcal{O}(d\,n \log n)$, where d is the decomposition depth of the motorcycle graph.

1 Introduction

The straight skeleton of a simple polygon was introduced to computational geometry about two decades ago in [1]. It is defined as the trace of the vertices as the polygon shrinks by moving its edges in a self-parallel manner towards the interior of the polygon. During this offsetting process edges disappear (so-called edge events, see the dotted offset in Figure 1 where the edge e disappears), and reflex vertices may run into other edges (so-called split events, see the dashed offset in Figure 1 where the vertex r runs into some edge and splits it). For more detailed information see, for example, the section on straight skeletons in [2].

Initially, a simple priority queue algorithm with a running time of $\mathcal{O}(n^2 \log n)$ has been proposed, simulating the shrinking process by computing all edge and split events. A theoretical breakthrough was the first sub-quadratic algorithm, by Eppstein and Erickson in [5] who also introduce the motorcycle graph. This graph consists of straight traces of 'motorcycles' that start at reflex vertices, with the speed and direction of the reflex vertex during the shrinking process; see Figure 2 for an example. A motorcycle's trace stops when it hits either the (already existing) trace of another motorcycle or the boundary of the polygon (as in [3] we assume that no two motorcycles collide). It seems that the motorcycle graph encodes essential information needed for the construction of the straight skeleton; several of the more recent algorithms for the straight skeleton build upon it. However, even the

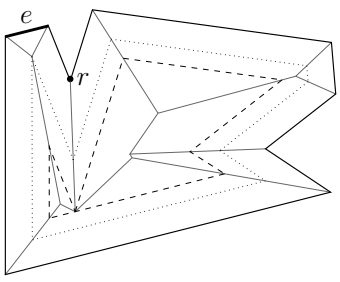

Figure 1 Straight skeleton offsetting process with edge and split events.

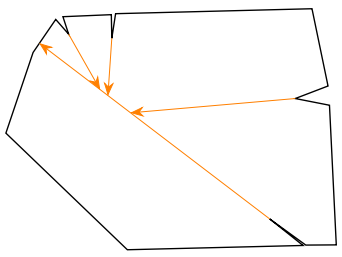

Figure 2 The motorcycle graph partitions a nonconvex polygon into convex cells.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

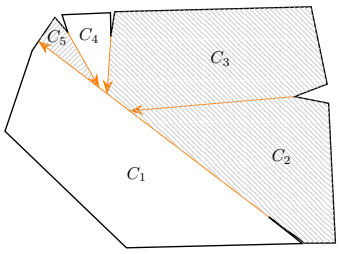

Figure 3 Motorcycle regions C_5 (motorcycle cell) and $C_2 \cup C_3$ (union of two regions).

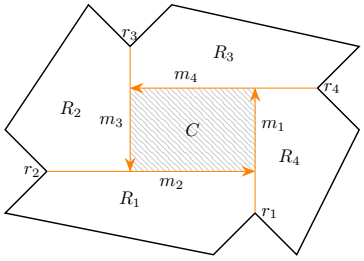

Figure 4 An inner motorcycle cell C, bounded by a cycle of dominant motorcycles.

derivation of the straight skeleton from the motorcycle graph has remained a complicated task being hard to implement. Also, the computation of the motorcycle graph itself is a challenging problem (the best known algorithm has a running time of $\mathcal{O}(n^{4/3+\epsilon})$ [6]).

The global nature of split events complicates the design of efficient *divide-and-conquer* algorithms for the straight skeleton. It took almost two decades (and several failing attempts) until the first correct and efficient algorithm of this type was published, in Cheng et al. [3]. Interestingly, this algorithm shows the currently best theoretical running time, $\mathcal{O}(n \log n \log r + r^{4/3+\varepsilon})$, for a polygon with n edges and r reflex vertices.

Here and later on, let $\mathcal{P}$ denote a simple n-vertex polygon, $\mathcal{M}(\mathcal{P})$ its motorcycle graph, and $\mathcal{S}(\mathcal{P})$ its straight skeleton (or just skeleton for brevity). The line segments forming $\mathcal{S}(\mathcal{P})$ will be called arcs.

In this note, we present a very simple divide-and-conquer algorithm that computes $\mathcal{S}(\mathcal{P})$ once $\mathcal{M}(\mathcal{P})$ is given. The idea is to divide $\mathcal{P}$ into cells according to $\mathcal{M}(\mathcal{P})$, to compute the skeletons of the motorcycle cells separately, and then merge them into $\mathcal{S}(\mathcal{P})$. We start in Section 2 by defining motorcycle regions and related concepts. The skeletons of regions are the topic of Section 3, in particular the relationship between the skeleton of a region and the skeletons of its subregions. The results from Section 3 are put to use in Section 4 for a divide-and-conquer algorithm that computes $\mathcal{S}(\mathcal{P})$. The divide step is trivial provided $\mathcal{M}(\mathcal{P})$ is available. The conquer step is as simple as Chew's [4] incremental method for the medial axis of a convex polygon. We first describe the algorithm for motorcycle graphs that are free of cycles. General motorcycle graphs are handled in Section 5, along with runtime considerations depending on the structure of these graphs. If the structure of $\mathcal{M}(\mathcal{P})$ is nice enough, the expected running time is $\mathcal{O}(n \log^2 n)$, while in the general case it is $\mathcal{O}(n^2 \log n)$.

2 Motorcycle regions

The motorcycle graph $\mathcal{M}(\mathcal{P})$ partitions $\mathcal{P}$ into polygons called *cells*, which are convex because a motorcycle egde emanates from each reflex vertex of $\mathcal{P}$, bisecting its interior angle. We first assume that each cell is supported by edges of $\mathcal{P}$. (Inner motorcycle cells, which are caused by cycles in $\mathcal{M}(\mathcal{P})$, will be handled later.) To combine motorcycle cells we introduce motorcycle regions. A *motorcycle region*, R, of $\mathcal{M}(\mathcal{P})$ is either a motorcycle cell defined by $\mathcal{M}(\mathcal{P})$, or $R = R_1 \cup R_2$ where R_1 and R_2 are regions sharing some motorcycle edge. See Figure 3 for an illustration of both kinds of regions.

From now on let $R = R_1 \cup R_2$ be a region of $\mathcal{M}(\mathcal{P})$, let m be the common motorcycle edge of R_1 and R_2, and denote with r the reflex vertex that defines m. To associate R with a skeleton suitable for future merging steps, a so-called *region polygon* for R (which is a superset of R and actually generates the desired skeleton) is needed. Following the recursive

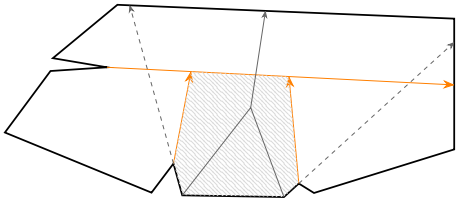

Figure 5 Motorcycle cell polygon (dashed, open) using only the cell's polygon edges.

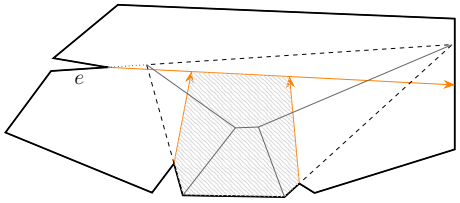

Figure 6 Motorcycle cell polygon (dashed) with the dominating edge e.

definition of motorcycle regions, we will define the region polygon for a motorcycle cell first, and then show how the region polygons for R_1 and R_2 are combined to obtain the region polygon for R.

Consider some motorcycle cell C. Each edge of C is either supported by an edge of $\mathcal{P}$ or by an edge of $\mathcal{M}(\mathcal{P})$. The former type of edges could be used to construct a (possibly unbounded) convex polygon by extending them until adjacent edges intersect; see Figure 5 for an example. However, merging two such polygons can create a polygon whose skeleton contains an arc *longer* than its motorcycle edge in $\mathcal{M}(\mathcal{P})$. This leads to complications during the merging process.

The solution is to also add edges of $\mathcal{P}$ that define motorcycle edges supporting C. Let m be a motorcycle edge supporting C, and let r be its reflex vertex in $\mathcal{P}$. The edge e of $\mathcal{P}$ incident to r and on the same side of m as C is called a *dominating edge* of C. Combining all dominating edges of C with all edges of $\mathcal{P}$ that support C still results in a convex polygon, the *motorcycle cell polygon*; see Figure 6 for an example. Note that the combined size of all motorcycle cell polygons is $\mathcal{O}(n)$.

In the following, let $\mathcal{R}$, $\mathcal{R}_1$ and $\mathcal{R}_2$ be the region polygons of R, R_1 and R_2, respectively. To obtain $\mathcal{R}$, the polygons $\mathcal{R}_1$ and $\mathcal{R}_2$ need to be merged at both endpoints, say r and v, of m. At the reflex vertex r this is done by simply truncating the respective edges of $\mathcal{R}_1$ and $\mathcal{R}_2$. Concerning v, this endpoint either lies on a polygon edge (which, therefore, is already part of the subregions' polygons), or on another motorcycle edge m' that dominates m, and thus the corresponding dominating edge is already part of the subregions' polygons. As a consequence, the merging of $\mathcal{R}_1$ and $\mathcal{R}_2$ at v is simple as well. Note that r is the only new reflex vertex that gets introduced by the merge.

3 Straight skeletons of motorcycle regions

The skeleton of a region R of $\mathcal{M}(\mathcal{P})$ is defined as the part of the skeleton of R's region polygon that lies within R. That is, $\mathcal{S}(R) = \mathcal{S}(\mathcal{R}) \cap R$.

As an important property of $\mathcal{S}(\mathcal{R})$, the skeleton arcs of reflex vertices are contained in R. Therefore, such skeleton arcs cannot cross region boundaries during the merge process.

▶ **Lemma 3.1.** *Let u be the arc of a reflex vertex r in $\mathcal{S}(\mathcal{R})$. Then $u \subset R$.*

Proof. Vertex r is part of R, therefore its motorcycle edge m is completely contained in R. If m hits the boundary of $\mathcal{P}$, then the hit edge of $\mathcal{P}$ is also part of R, and thus the arc u is contained in R. Assume now that m is blocked by another motorcycle edge m'. The motorcycle cells that have parts of both m and m' on their boundary are part of R. Therefore the associated dominating edge of m' with respect to these cells contributes to the boundary of $\mathcal{R}$. Thus u cannot cross m', and $u \subset R$ again. ◀

Let us now look in more detail at the straight skeleton within a single motorcycle cell C. We say that an edge e of $\mathcal{P}$ is *relevant* for C, if the unique face $f_{\mathcal{P}}(e)$ that e defines in $\mathcal{S}(\mathcal{P})$ has a nonempty intersection with C.

▶ **Theorem 3.2.** *The relevant edges for C (extended if necessary) form a convex polygon in the cyclic order given by $\mathcal{P}$.*

Proof. Clearly, the edges of $\mathcal{P}$ that support C are all relevant for C, and they form a convex polygon. All other edges relevant for C must cross a motorcycle edge during the shrinking process to have a part of C in their face in $\mathcal{S}(\mathcal{P})$. Therefore, they must form convex angles with the adjacent edges in the polygon formed by all relevant edges; the arcs of reflex vertices are shorter than their corresponding motorcycle edges. ◀

▶ **Corollary 3.3.** *Let e be an edge of $\mathcal{R}_2$ that is not an edge of $\mathcal{R}_1$, and such that its face $f_R(e)$ in the straight skeleton of $R = R_1 \cup R_2$ has a nonempty intersection with R_1. Then e forms convex angles when inserted into $\mathcal{R}_1$.*

Proof. For each motorcycle cell $C \subseteq R_1$ with $f_R(e) \cap C \neq \emptyset$, we know from Theorem 3.2 that e is in *convex position* with C's motorcycle cell polygon, that is, e cuts off a single convex vertex from this polygon. Therefore e forms convex angles with its adjacent edges in $\mathcal{R}_1$. ◀

The straight skeleton behaves nicely when inserting an edge in convex position. During the shrinking process, the new edge is always ahead of the parts of the adjacent edges that were cut off. All other edges remain unchanged. Thus the skeleton faces of old edges can shrink but never expand.

For the merge of $\mathcal{S}(\mathcal{R}_1)$ and $\mathcal{S}(\mathcal{R}_2)$ we need to find the edges from one region that can influence the skeleton of the other region within $\mathcal{S}(\mathcal{R})$. The following lemma gives a necessary condition for such edges.

▶ **Lemma 3.4.** *Only edges of $\mathcal{R}_2$ whose faces in $\mathcal{S}(R_2)$ are bounded by the common motorcycle edge m can change $\mathcal{S}(\mathcal{R}_1)$ within $\mathcal{S}(R)$, the merged skeleton.*

Proof. Let e be an edge of $\mathcal{R}_2$ whose face $f_R(e)$ in $\mathcal{S}(R)$ has a nonempty intersection with R_1. Then $f_R(e)$ must be intersected by m, since e is on the opposite side of m. Edges of R_1 with faces that have nonempty intersection with R_2 in $\mathcal{S}(R)$ are in convex position with respect to $\mathcal{R}_2$, by Theorem 3.3. Consequently, e's face in $\mathcal{S}(R_2)$ can only shrink; more precisely, $f_R(e) \cap R_2 \subseteq f_{R_2}(e)$. But this implies that $f_{R_2}(e)$ is bounded by m. ◀

The following related lemma is stated without proof, due to lack of space.

▶ **Lemma 3.5.** *Let E be the (cyclically ordered) set of edges of $\mathcal{R}_2$ whose faces in $\mathcal{S}(R_2)$ are bounded by m, excluding the edge adjacent to m's reflex vertex r in R_2. Then E forms a convex chain, and its edges form convex angles when inserted into $\mathcal{R}_1$.*

4 Divide-and-conquer

We are now ready to describe our divide-and-conquer algorithm. In the divide part, we use the motorcycle graph $\mathcal{M}(\mathcal{P})$ to recursively divide $\mathcal{P}$ along motorcycle edges. Since we assumed $\mathcal{M}(\mathcal{P})$ to have no cycles, there exists a motorcycle edge m that hits the boundary of $\mathcal{P}$. This divides $\mathcal{P}$ into two polygons, P_1 and P_2. This dividing step can be repeated, using motorcycle edges m_1 and m_2 that split P_1 and P_2 into two parts, respectively, and is

iterated until all motorcycle edges have been used and the final polygons are the motorcycle cells of $\mathcal{M}(\mathcal{P})$. The division process can be represented in a tree, which inherits the future merge plan and is therefore called the *merge tree*.

In the conquer part of our algorithm, the results from the previous section are put to use. Suppose $\mathcal{S}(\mathcal{R}_1)$ and $\mathcal{S}(\mathcal{R}_2)$ have already been computed, and need to be merged into $\mathcal{S}(\mathcal{R})$. For $\mathcal{S}(R_1)$, all edges with faces bounded by m are computed. By Lemma 3.5, these edges form a convex chain E_1 that forms convex angles when inserted into $\mathcal{R}_2$. Inserting E_1 into $\mathcal{S}(\mathcal{R}_2)$ results in $\mathcal{S}(\mathcal{R}_2 \cup E_1)$. The same is done with the roles of R_1 and R_2 exchanged, to obtain $\mathcal{S}(\mathcal{R}_1 \cup E_2)$. Note that both skeletons coincide along m, and only need to be glued together along m to obtain $\mathcal{S}(\mathcal{R})$.

Still missing is the part of the algorithm that updates a straight skeleton during the insertion of a convex chain E. We adapt the approach from Chew [4] that computes the medial axis of a convex polygon (which equals its straight skeleton). We insert the edges of E in random order, with each update taking time proportional to the number of arcs of the inserted edge's face as its boundary is traced out.

Suppose an edge e in convex position is inserted into a region polygon and its skeleton needs to be updated. The expected number of arcs defined by e and other (already inserted) edges from E is still constant (here Chew's analysis still applies), but the expected number of arcs defined by e and edges from the original polygon (so-called *mixed arcs*) is not constant. Let l be the size of the original polygon, k the size of E, and $n = l+k$. Suppose that $i-1$ edges have already been inserted and the skeleton of the resulting polygon computed. Inserting the i^{th} edge may cause the construction of new mixed arcs. The maximal number of mixed arcs after the insertion is $l+i-1$, and since the insertion order is randomized, the expected number of mixed arcs supported by the new edge is $(l+i-1)/i$. Summing over all insertions gives the expected total number of mixed arcs when E is inserted: $\sum_{i=1}^{k}(l+i-1)/i = \mathcal{O}(n \log n)$.

As a conseqence, the expected running time for all merges on a single level of the merge tree is $\mathcal{O}(n \log n)$. The running time of the complete algorithm depends on the depth of the merge tree and we get the following theorem:

▶ **Theorem 4.1.** *Let d be the height of the merge tree. Then the straight skeleton of $\mathcal{P}$ can be computed in $\mathcal{O}(d\ n \log n)$ expected time.*

5 General motorcycle graphs and runtime considerations

Now we lift the restriction that the motorcycle graph must be free of cycles. A motorcycle graph cycle is created by a cycle of dominating motorcycles bounding an inner region (see Figure 4). Merges cannot be performed as above, since no two regions share a complete motorcycle edge. However, it is possible to first compute the skeleton of the inner cell, and use the known merge procedure for the skeletons of the outer regions.

Let C be an inner motorcycle cell, bounded by the cycle of dominating motorcycle edges $m_1, \ldots, m_k$. Let R_i denote the region bounded by m_i and m_{i+1} (indices modulo k).

Now consider an edge e of $\mathcal{R}_i$, such that its face $f_{\mathcal{P}}(e)$ in $\mathcal{S}(\mathcal{P})$ has a nonempty intersection with C. Then the face of e in $\mathcal{S}(R_i)$ must be bounded by m_{i+1}, giving a necessary condition for an edge to have a face in $\mathcal{S}(\mathcal{P})$ intersecting with C. It can be shown that all such edges form a convex polygon, enabling the computation of $\mathcal{S}(C)$ in linear time.

Having computed $\mathcal{S}(C)$, it is now possible to detect the edges that sweep over C during the shrinking process, and thus need to be included in the skeletons of the outer regions adjacent to C. Merging all these edges into the skeletons of the appropriate regions R_i can be done in overall $\mathcal{O}(l \log l)$ expected time, with l being the number of edges involved.

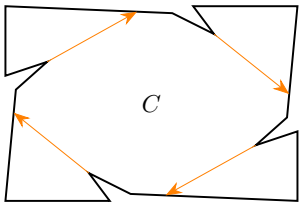

Figure 7 All ears are merged into C.

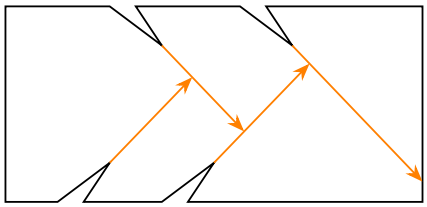

Figure 8 The merge tree degenerates to a path.

Finally, the updated skeletons of the outer regions are combined. First, these skeletons can be restricted to their region (remember that the skeleton of C does not change). Then they can be merged using the original merge procedure along a common motorcycle edge. Balanced binary merges can be used to get an overall expected running time of $\mathcal{O}(n \log^2 n)$.

Whereas motorcycle graphs with cycles can be integrated into the divide step without causing the running time to increase, the dependency on the height of the merge tree (Theorem 4.1) can cause an expected running time of $\mathcal{O}(n^2 \log n)$. There are two kinds of motorcycle graph structures producing merge trees with linear height. The first one results from a 'central' motorcycle cell C with linearly many adjacent cells that need to be merged with C as they have no common motorcycle edge with another cell (see Figure 7 for an example). The second kind occurs when for linearly many motorcycles $m_1, \ldots, m_k$, the motorcycle m_i crashes into the trace of m_{i+1}, and m_k hits the polygon boundary as in Figure 8. For both structures, there are efficient solutions once they have been identified in the motorcycle graph. However, detecting these structures efficiently is an open problem.

6 Conclusions

We have presented a simple and easy-to-implement divide-and-conquer algorithm that derives the straight skeleton of a polygon from its motorcycle graph. The running time depends on the structure of the motorcycle graph, and is expected $\mathcal{O}(n \log^2 n)$ if the motorcycle graph is reasonably 'balanced', competing with the best known algorithm [3]. Still, it seems to be a long way till such a running time may be achieved for constructing the straight skeleton from scratch, as precomputing the motorcycle graph is believed to be the hardest part.

References

1 Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A novel type of skeleton for polygons. In *J.UCS The Journal of Universal Computer Science*, pages 752–761. Springer, 1996.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Co., Inc., 2013.

3 Siu-Wing Cheng, Liam Mencel, and Antoine Vigneron. A faster algorithm for computing straight skeletons. In *European Symposium on Algorithms*, pages 272–283. Springer, 2014.

4 L. Paul Chew. Building voronoi diagrams for convex polygons in linear expected time. Technical report, 1990.

5 David Eppstein and Jeff Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999.

6 Antoine Vigneron and Lie Yan. A faster algorithm for computing motorcycle graphs. *Discrete & Computational Geometry*, 52(3):492–514, 2014.