Christopher TSCHERNE, BSc.

Straight Skeletons & Motorcycle Graphs

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematical Computer Science

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Franz AURENHAMMER

Institute for Theoretical Computer Science

Graz, May 2018

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Abstract

The straight skeleton is a skeletal structure for polygons that results from a self-parallel shrinking process. Compared to the medial axis it only contains straight line segments and has a lower combinatorial complexity. There are many uses for the straight skeleton in computer science.

In 2014 Cheng, Mencel and Vigneron [10] presented an algorithm for computing the straight skeleton of a polygon. This algorithm runs in $\mathcal{O}(n \log(n) \log(r) + r^{\frac{4}{3}+\varepsilon})$ time for polygons with n vertices and r reflex vertices. This is currently the best known theoretical running time. However, this algorithm is quite involved and no implementation was provided to show its practical usefulness.

This thesis presents a practical divide-and-conquer algorithm [1] for computing the straight skeleton of a simple polygon based on motorcycle graphs that runs in $\mathcal{O}(d\, n \log(n))$ expected time where d is the height of the merge tree. In the best case, the merge tree has a height of $\mathcal{O}(\log(n))$ and a linear height in the worst case. Furthermore, an implementation of this algorithm is presented to show its practical usefulness.

Contents

List of Figures

1 Introduction

1.1 Motivation and Problem Statement

A skeleton can be interpreted as a thinner and more compact representation of an object that still shows geometrical and topological features of the original object.

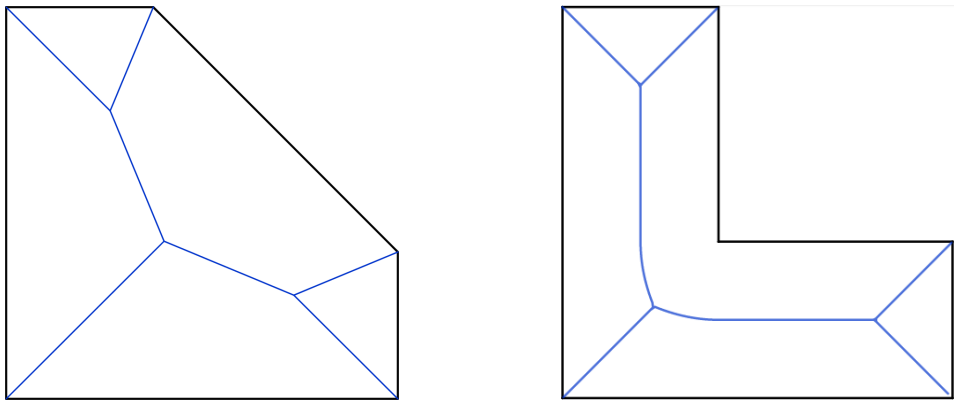
In biology, skeletons are the supporting structures of many organisms. The skeleton of a human being for example shows these features even though it is a simpler version of the whole body. Features as height, connectivity between body parts and deformations can be reconstructed from the skeleton. In addition, the shape and species of the living being can be distinguished between skeletons of the same size - a bird and a cat have very distinct skeletons.

In mathematics and computer science, the notion of skeletons for polygons is similar to skeletons in biology. Skeletons contain relevant information about the polygon such as size and shape.

1.1.1 Medial Axis

The medial axis of a polygon is one of the most well-known skeletal structures. It is defined as the set of all points that have no unique closest point on the boundary of the polygon. An equivalent definition is given by the set of all centers of maximal circles inside the polygon.

For convex polygons the medial axis always consists of straight line segments. On the other hand, the medial axis contains parabolic curves for non-convex polygons.



(a) Convex polygon **(b)** Non-convex polygon

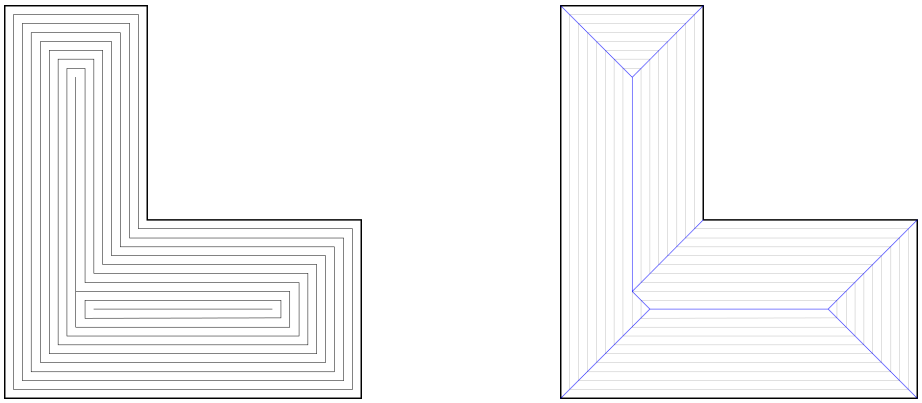
Figure 1.1: Medial axis

1.1.2 Straight Skeletons

The straight skeleton is another skeletal structure for polygons and was first introduced by Aichholzer et al. (1995) [2].

In comparison to other skeletons like the medial axis, it only contains straight line segments even for non-convex polygons. This structure is not defined by a distance function like the medial axis but uses a continuous shrinking process to create event points which are connected by straight line segments (Details for the straight skeleton are explained in Chapter 2).

The simple structure of the straight skeleton might suggest an easy solution for its computation. However, the continuous shrinking process makes it hard to design fast algorithms for its computation.



(a) Offset polygons **(b)** Straight skeleton (blue)

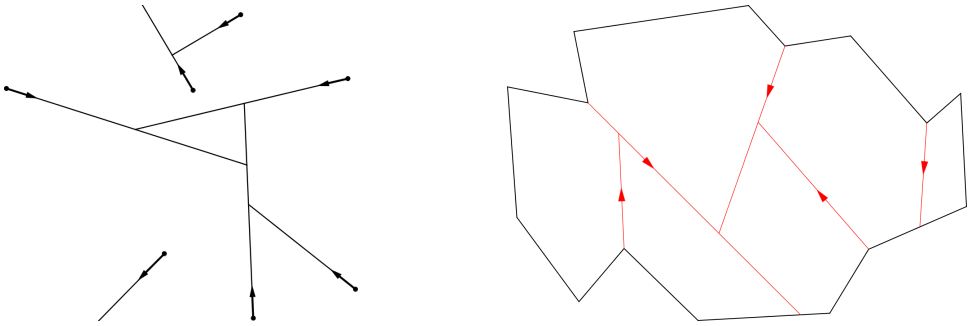
Figure 1.2: Shrinking process

1.1.3 Motorcycle Graphs

Imagine motorcycles driving on a plane on different directions represented by straight lines. Every motorcyclist is dropping dangerous spikes behind them, causing other motorcycles to crash on the spot when they cross the path of another motorcyclist.

After all motorcycles are either crashed or the motorcyclists ensured that they escaped into safety, their paths will form a graph like the one in Figure 1.3a. This graph is called the motorcycle graph based on the idea of the crashing motorcycles.

This concept can also be used for polygons. Motorcycles start at vertices of the polygon that have an interior angle larger than 180°. In addition, they also crash when they hit the boundary of the polygon. The resulting graph is called induced motorcycle graph or motorcycle graph of the polygon (Figure 1.3b).



(a) General motorcycle graph **(b)** Motorcycle graph of a polygon

Figure 1.3: Motorcycle graph

This graph can be used as a tool for the computation of the straight skeleton because it contains information about the interaction of the non-convex vertices during the shrinking process.

1.2 Outline of the Thesis

Chapter 2 introduces formal definitions for the straight skeleton and motorcycle graphs and gives basic properties and relevant notations.

Chapter 3 gives an overview of previous works on straight skeletons and motorcycle graphs and discusses algorithms for finding these structures. Furthermore, related topics are presented.

Chapter 4 presents a new divide-and-conquer algorithm for finding the straight skeleton with the help of motorcycle graphs.

In Chapter 5 an implementation of the algorithm is presented with further details.

Chapter 6 shows experimental results for different types of polygons that were achieved with the implementation of the algorithm.

Chapter 7 summarizes the results of this thesis. Furthermore, possible improvements are discussed.

2 Straight Skeletons & Motorcycle Graphs

This chapter presents a formal definition for both the straight skeleton and the motorcycle graph. Furthermore, relevant properties and basics are introduced that will be important during later chapters.

2.1 Basics & Notation

Notation (Polygon)**.** *A polygon is a plane shape (two-dimensional) bounded by connected straight line segments. These line segments are called edges. The points where two edges meet are called vertices (singular: vertex).*

Definition 2.1 (Simple Polygon)**.** *A polygon is called simple if all intersections of two edges are the vertices of the polygon and each vertex lies on exactly two edges. Otherwise, it is called self-intersecting, non-simple or complex.*

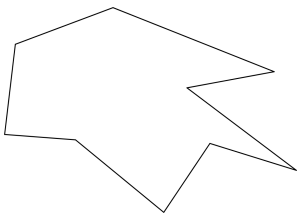


Figure 2.1: Simple polygon

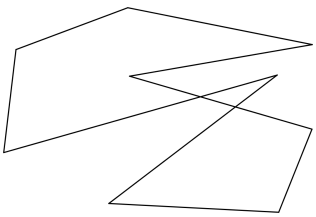


Figure 2.2: Complex polygon

Definition 2.2 (Boundary, Interior)**.** *Let P be a simple polygon. The boundary of P is defined as $\partial(P) = \{x \in \mathbb{R}^2 \mid \forall \varepsilon > 0\, \exists y_1 \in P, y_2 \notin P : y_1, y_2 \in B_\varepsilon(x)\}$ where $B_\varepsilon(x) = \{y \in \mathbb{R}^2 \mid \|x-y\| < \varepsilon\}$. The interior of P is defined as $int(P) = P \setminus \partial(P)$.*

Definition 2.3 (Hole)**.** *Let P' and P_i, $i \in \{1, \ldots, k\}$, be simple polygons. A polygon P is called polygon with k holes if $P = P' \setminus \bigcup_{i=1}^{k} int(P_i)$ with $\bigcup_{i \in I} P_i \subset int(P')$ and $P_i \cap P_j = \emptyset, i \neq j$. The polygons P_i are called holes.*

Remark. *In this thesis all polygons are simple polygons without holes in the plane if not mentioned otherwise.*

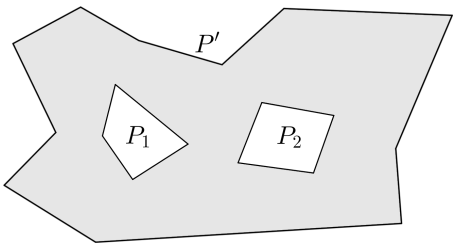


Figure 2.3: Polygon (gray) with two holes

Definition 2.4 (Reflex Vertex)**.** *A vertex of a polygon is called reflex if its interior angle is larger than π radians* (180°)*. Otherwise, it is called convex.*

Remark. *In this thesis, n denotes the number of vertices of a polygon and r denotes the number of reflex vertices.*

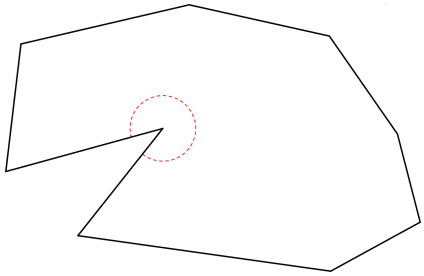


Figure 2.4: Polygon with one reflex vertex

Remark. *Edges that meet in a vertex with an angle of π radians can be merged into one edge without changing the skeleton. Therefore, w.l.o.g., vertices never have an interior angle of π radians.*

Notation (Bisector)**.** *The angular bisector of two edges x and y is denoted by $bis(x, y)$.*

Remark. *Edges x and y will not always be connected and the bisector of two lines is not uniquely defined. $bis(x, y)$ is the bisector that is relevant inside the polygon.*

Definition 2.5. *For an edge e of a polygon P, l_e denotes the line supported by e.*

Definition 2.6. *The intersection of two lines (or edges) a, b is denoted by $a \cap b$. $a \cap b$ is well-defined for lines if and only if a and b are not parallel. $a \cap b$ is well-defined for edges if and only if the line segments created by a and b have an intersection.*

2.2 Straight Skeletons

The straight skeleton is a skeletal structure similar to the well-known medial axis. Both structures are even identical for convex polygons. However, the medial axis may contain parabolic curves, whereas the straight skeleton is only made up of straight line segments, namely, the angular bisectors.

The medial axis for a polygon can be defined as the set of all points that do not have a unique closest point on the boundary of the polygon. While the medial axis can be defined using a distance function, the straight skeleton is defined by a continuous shrinking process. The straight skeleton for simple polygons and the shrinking process were first introduced by Aichholzer et al. (1995) [2].

Imagine that the edges of a polygon P are moving inwards in a self-parallel manner at a certain speed. The vertices are moving along the bisector of its adjacent edges if the edges have the same speed. The resulting polygons are called offset polygons. An example is shown in Figure 2.5

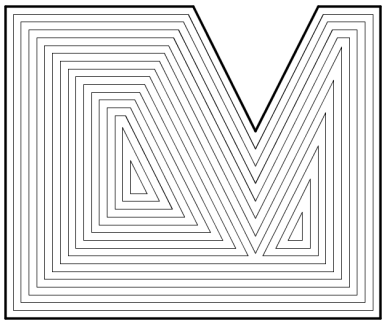
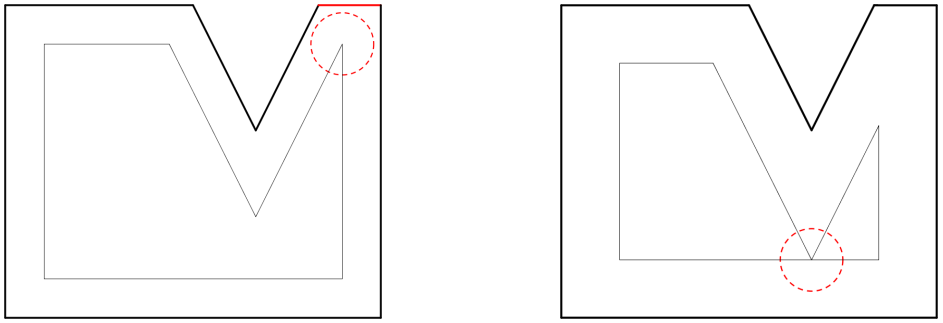


Figure 2.5: Offset polygons

This process continues until one of the following events changes the topology of the polygon:

1. **Edge event**: Happens when an edge shrinks until it vanishes (Figure 2.6a).
2. **Split event**: This event occurs when a reflex vertex hits another edge of the polygon. This splits the polygon into two parts and the shrinking process continues for each of them (Figure 2.6b).



(a) Edge event **(b)** Split event

Figure 2.6: Events during the shrinking process

The shrinking process continues for the resulting polygons until they vanish themselves.

Definition 2.7 (Straight Skeleton [2])**.** *Let P be a polygon. The straight skeleton $S(P)$ is defined as the union of the pieces of angular bisectors traced out by polygon vertices during the shrinking process described above.*

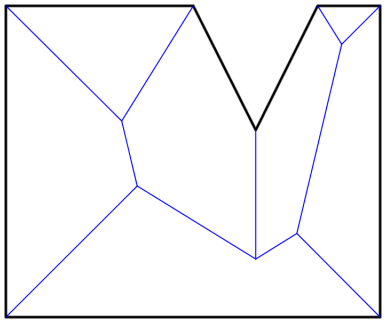
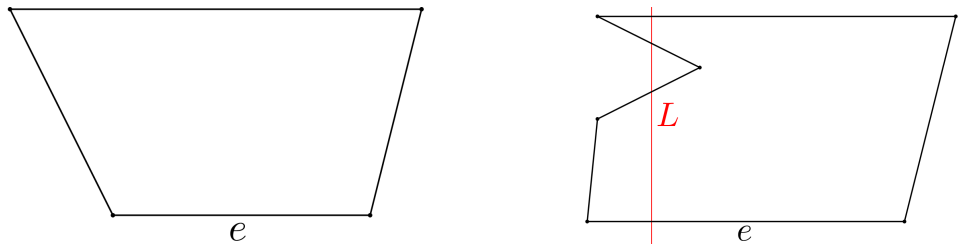


Figure 2.7: Straight skeleton

The bisector segments are called *arcs* and their endpoints (ignoring points on the polygon) are called *nodes* of $S(P)$. The straight skeleton divides the polygon in regions which are called *faces*. Each edge e is connected to exactly one face which is denoted by $f(e)$.

2.2.1 Properties

Definition 2.8 (Monotonicity)**.** *A polygon P is called monotone with respect to an edge e, if every line L normal to e intersects the boundary of P at most twice.*



(a) Monotone **(b)** Not monotone

Figure 2.8: Monotonicity with respect to the edge e

Lemma 2.9. *(Aichholzer et al. (1995) [2]) A face $f(e)$ created by the straight skeleton is monotone with respect to its defining edge e.*

Lemma 2.10. *(Aichholzer et al. (1995) [2]) The straight skeleton $S(P)$ of a polygon P with n vertices has the following properties:*

1. *$S(P)$ is a tree*
2. *$S(P)$ divides P into n connected faces*
3. *$S(P)$ has $n-2$ nodes*
4. *$S(P)$ has $2n-3$ arcs*

The geometry of the straight skeleton is simpler than the geometry of the medial axis because it only contains straight line segments. The straight skeleton has $2n-3$ arcs for a polygon P of size n with r reflex vertices. On the other hand, the medial axis has $2n+r-3$ arcs with r of them being parabolic curves.

The straight skeleton and the medial axis both represent the polygon in a compact way. However, the straight skeleton can be very sensitive to small changes (Figure 2.9). This is caused by a drastic speed changes for some motorcycle edges which can affect all other regions.

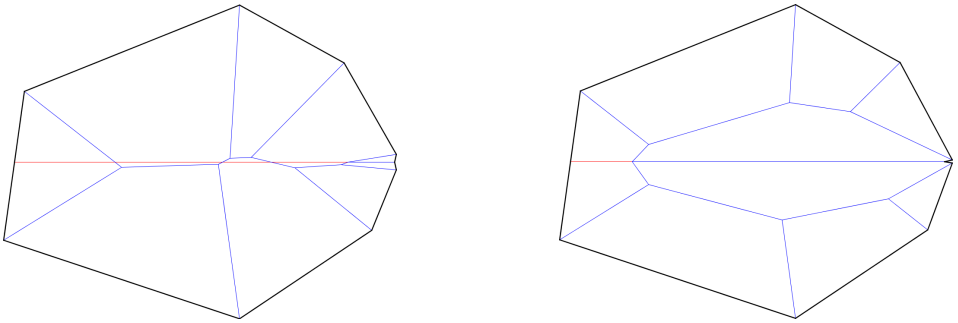


Figure 2.9: Small local changes affecting the straight skeleton

2.2.2 Computation

Aichholzer et al. (1995) [2] presented a direct method to compute the straight skeleton which results in an $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space algorithm. This approach considers each pair of edges to compute the next event. This algorithm can be modified by using a priority queue for the events which improves the runtime to $\mathcal{O}(n^2 \log(n))$ at the cost of $\mathcal{O}(n^2)$ storage.

Improvements for this rather slow runtime and related work will be reviewed in Chapter 3.

2.3 Bisector Graphs

$S(P)$ contains line segments supported by bisectors of polygon edges. Every node in $S(P)$ is the intersection point of three of these bisector pieces.

Definition 2.11 (Bisector Graph)**.** *Let P be a polygon. A straight line graph is called bisector graph $B(P)$ of P if*

1. *each line segment of $B(P)$ is a bisector piece of two edges of P.*
2. *every internal node has degree three and*
3. *there is a bijection of the vertices of P to all nodes of degree one.*

The bisector graph is not unique. A bisector graph can have self-intersections and parts of it can lie outside of the polygon (Figure 2.10). Therefore, a bisector graph does not necessarily define a proper face structure of the polygon P.

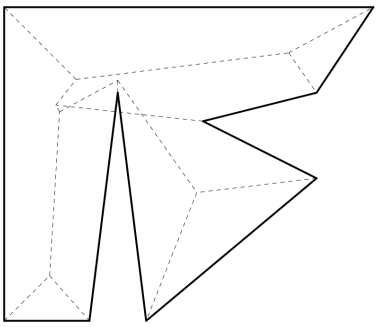


Figure 2.10: Self-intersecting bisector graph (Based on polygons from [2])

Thus, only plane (implies no self-intersection) and cycle-free bisector graphs that lie completely inside of the polygon are of interest. However, even that does not imply a unique structure.

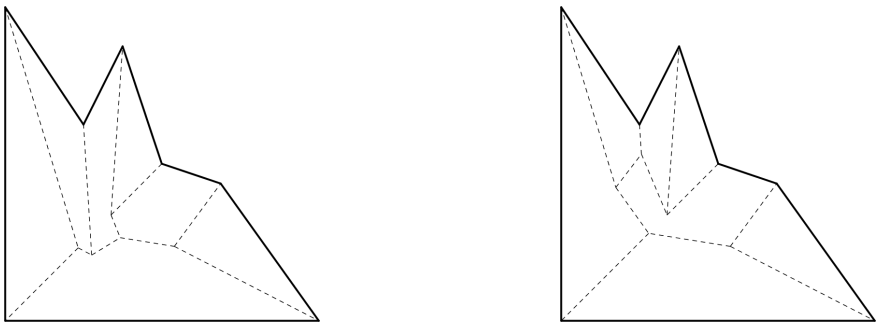


Figure 2.11: Different bisector graphs of the same polygon (Based on polygons from [2])

Roof Model

A plane bisector graph can be interpreted as a projection of a roof. For the straight skeleton this roof can be defined by the shrinking process. The polygon lies on a plane and the time indicates the height for the offset polygons.

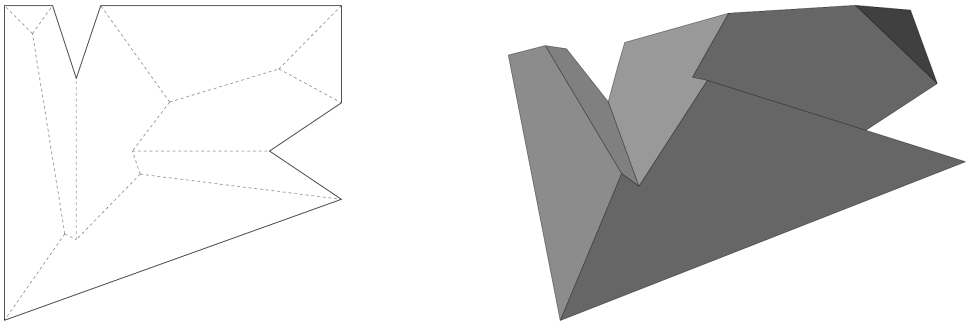


Figure 2.12: Roof model and corresponding straight skeleton (Input from [2])

Aichholzer et al. (1995) [2] showed that there is a bijection between roofs and plane bisector graphs and used roof models to prove basic properties of the straight skeleton.

2.4 Motorcycle Graphs

Definition 2.12 (Motorcycle)**.** *Let P be a polygon and R be the set of all reflex vertices of P. A motorcycle m_i, $i \in \{1, \ldots, |R|\}$, is a point starting at a reflex vertex $r_i \in R$ with velocity $v_i \in \mathbb{R}^+$ and moving in a direction d_i.*

Remark. *In this thesis, the direction d_i always coincides with the direction of the outgoing bisector of r_i. In general, a motorcycle does not have to move along the bisector of its reflex vertex.*

Definition 2.13 (Track)**.** *The path p_i of a motorcycle m_i is defined as the outgoing ray $\{r_i + \tau \cdot v_i \cdot d_i | \tau > 0\}$ of r_i with direction d_i. The track $p_{i,\tau}$ is the line segment from $p_i(0)$ to $p_i(\tau), \tau > 0$. A track ends if it collides with the track of another motorcycle m_j that reaches the collision point first or collides with an edge of P for increasing values of τ. Let t_i be the resulting track of the motorcycle m_i.*

Definition 2.14 (Motorcycle Graph)**.** *The motorcycle graph $\mathcal{M}(P)$ of a polygon P is defined as the arrangement of the line segments t_i. These line segments are also called motorcycle edges.*

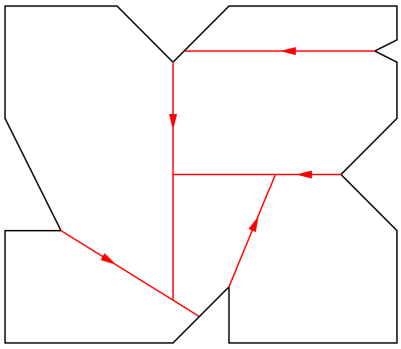


Figure 2.13: Motorcycle graph (red) with uniform velocities

Remark. *The motorcycle graph is empty for convex polygons and a motorcycle might never crash for an unbounded polygon.*

The velocity v_i of a motorcycle m_i can be arbitrary in general. However, there is a correlation between the straight skeleton and the motorcycle graph of a polygon P: Reflex vertices with an obtuse angle move slower during the shrinking process than reflex vertices with an acute angle. An example is shown in Figure 2.14.

(a) Obtuse angle **(b)** Acute Angle

Figure 2.14: Motorcycle graph velocities

Lemma 2.15 (Velocity)**.** *Let $\alpha \in (\pi, 2\pi)$ be the angle of the reflex vertex. Then the velocity v of the outgoing motorcycle during the shrinking process is given by*

$$v = \frac{1}{\sin\left(\frac{\alpha}{2}\right)} \in (1, \infty)$$

Proof. It needs to be shown that v is well-defined. This follows directly from the domain of α:

$$\begin{aligned} \alpha \in (\pi, 2\pi) \Leftrightarrow \frac{\alpha}{2} \in \left(0, \frac{\pi}{2}\right) \Rightarrow \sin\left(\frac{\alpha}{2}\right) \in (0, 1) \\ \Rightarrow \sin\left(\frac{\alpha}{2}\right) \neq 0. \end{aligned}$$

Let $t > 0$ be the current time during the shrinking process and β be the angle between l and t.

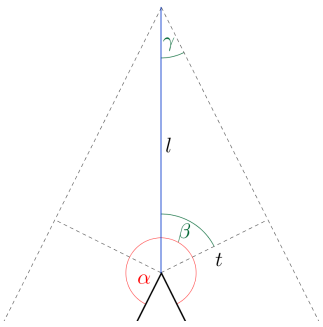


Figure 2.15: Geometric analysis of the velocity

The edges grow in a self-parallel manner and therefore β can be obtained by

$$\beta = \frac{1}{2}\left(\alpha - 2 \cdot \frac{\pi}{2}\right).$$

Using basic properties of angles it follows

$$\begin{aligned}\gamma &= \pi - \frac{\pi}{2} - \beta = \frac{\pi}{2} - \beta = \frac{\pi}{2} - \frac{1}{2}(\alpha - \pi) \\ &= \pi - \frac{\alpha}{2}.\end{aligned}$$

By using the law of sine it finally follows that

$$\frac{l}{\sin\frac{\pi}{2}} = \frac{t}{\sin\left(\pi - \frac{\alpha}{2}\right)} \Leftrightarrow l = t \cdot \frac{1}{\sin\left(\pi - \frac{\alpha}{2}\right)} \Leftrightarrow l = t \cdot \underbrace{\frac{1}{\sin\left(\frac{\alpha}{2}\right)}}_{v}$$

□

Definition 2.16 (Degenerate)**.** *A polygon is called degenerate if two motorcycles collide with each other at the same time.*

Remark. *In this thesis all polygons are non-degenerate polygons if not mentioned otherwise.*

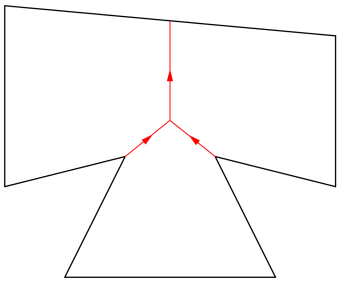


Figure 2.16: Degenerate polygon

Remark. *The motorcycle graph of a degenerate polygon is created by introducing a new reflex vertex at the collision position. The offset polygon at the time of the collision defines the angle of the reflex vertex and the properties of the corresponding motorcycle edge.*

Definition 2.17 (Dominance Relation)**.** *The line segment t_j dominates t_i ($t_j \succ t_i$) if the track of m_i collides with the track of m_j and t_i ends at the collision point.*

Definition 2.18 (Cycle)**.** *Let $T = \{t_i \mid i \in \{1, \ldots, k\}\}$ be a subset of size $k \in \mathbb{N}$ of line segments of $\mathcal{M}(P)$. T is called a cycle if there is a permutation $\pi \in \mathcal{S}_k$ with $t_{\pi(1)} \succ t_{\pi(2)} \succ \ldots \succ t_{\pi(k-1)} \succ t_{\pi(k)} \succ t_{\pi(1)}$*

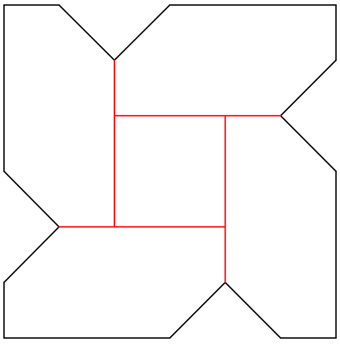
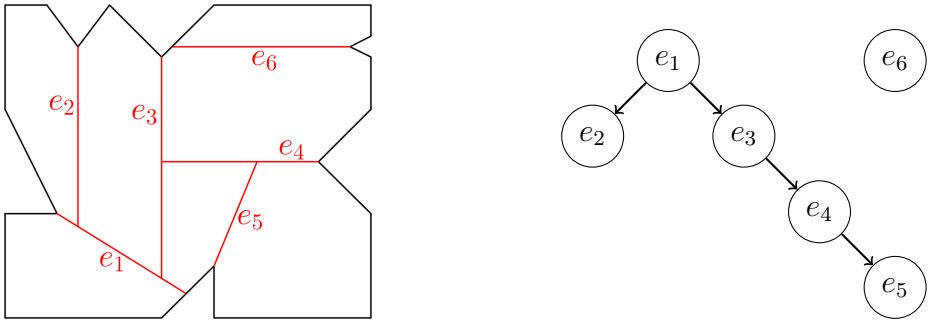


Figure 2.17: Cycle in a motorcycle graph

Definition 2.19 (Dominance Graph, Dominance Tree)**.** *Let P be a polygon and $\mathcal{M}(P)$ its motorcycle graph. The dominance relation $\succ$ induces a directed graph $T = (V, E)$ where V represents edges of $\mathcal{M}(P)$ and $E = \{(e, f) \mid e, f \in V, e \succ f\}$. T is called dominance graph. A component G of T is called dominance tree if the defining edges of G do not contain a cycle in $\mathcal{M}(P)$.*



(a) Motorcycle graph **(b)** Dominance trees

Figure 2.18: Dominance graph

Remark. *If $M(P)$ does not contain a cycle then each component of the dominance graph is a directed, rooted tree and the underlying undirected graph of a dominance graph is a forest.*

Definition 2.20 (Motorcycle Cell)**.** *The motorcycle graph $\mathcal{M}(P)$ partitions the polygon P into subpolygons. These subpolygons are called motorcycle cells.*

Remark. *A motorcycle cell is convex.*

Proof. Let C be a motorcycle cell and let $e(C)$ be the set of edges of the polygon that bound C. Chains of edges in $e(C)$ are in convex position, otherwise C would

contain a reflex vertex. Motorcycle edges are in convex position with the adjacent edges of their defining reflex vertex because the motorcycle edge divides the interior angle into two equal parts. Furthermore, motorcycle edges either crash into other motorcycle edges or into polygon edges. In both scenarios, only convex angles are created.

Therefore, all interior angles of a motorcycle cell are convex. □

Remark. *If $\mathcal{M}(P)$ is empty, then P is the only motorcycle cell induced by $\mathcal{M}(P)$. A motorcycle cell is bounded by edges of the motorcycle graph and edges of P. These edges are in convex position because the motorcycle cell itself is convex.*

Definition 2.21 (Induced Polygon)**.** *Let C be a motorcycle cell and let $e(C)$ be the set of edges of the polygon that bound C. $\mathcal{H}_{l_e}$ denotes the closed half-plane defined by $l_e, e \in e(C)$, such that $C \subseteq \mathcal{H}_{l_e}$. The convex polygon $P_C = \bigcap_{e \in e(C)} \mathcal{H}_{l_e}$ is called the induced polygon of the motorcycle cell C.*

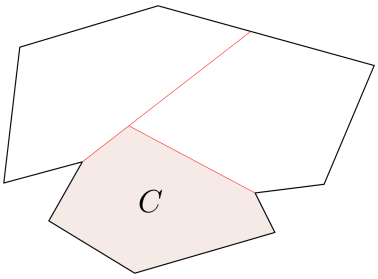


Figure 2.19: Motorcycle cell

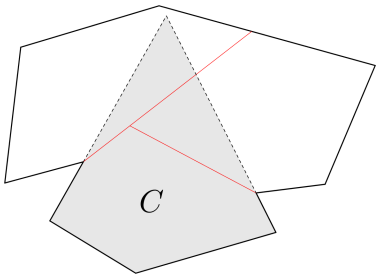


Figure 2.20: Induced polygon

Remark. *Induced polygons are well-defined because the intersection is not empty due to $C \subseteq P_C$ and the intersection of half-planes is convex.*

The induced polygons might be unbounded and it is also possible that an induced polygon is only defined by a single edge. Cycles are another special case. The motorcycle graph does not induce a special polygon for the closed region in the center of a cycle but induces polygons normally everywhere else.

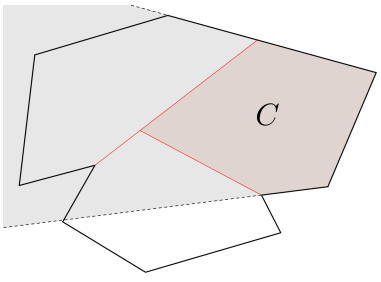


Figure 2.21: Induced polygon (unbounded)

Definition 2.22 (Region)**.** *A (motorcycle) region is either a motorcycle cell or results from merging (two) regions that share an entire motorcycle edge.*

The edges of a region also define a polygon. However, this polygon is not convex anymore and is defined as a polygonal chain (might be open) constructed by using the two involved region polygons. This chain starts at the reflex vertex of the shared motorcycle edge and follows all edges of one polygon and then follows the edges of the other polygon until it meets the reflex vertex again. The resulting polygonal chain can have multiple open sections and can also have crossings which will be covered in Section 4.2.4.

For cycles a new region results from merging all involved regions and the center region together instead of just merging two regions.

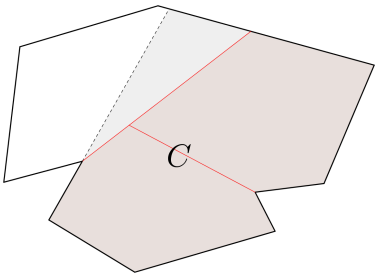


Figure 2.22: Region and region polygon

Lemma 2.23. *There are $r + 1$ motorcycle cells.*

Proof. Let $\mathcal{M}(P)$ be cycle-free. There is one cell if there are no reflex vertices. Let $|\mathcal{M}(P)| \geq 1$. Insert each line segment of $\mathcal{M}(P)$ in breadth-first order with respect to the dominance graph starting with the root of each component. Every line splits one region into two other regions. This creates $r + 1$ cells.

Every cycle of size k in $\mathcal{M}(P)$ splits one region into k new ones that are connected to the boundary of P and creates one extra motorcycle cell in the center. The number of cells increases by k and therefore by one for each motorcycle edge in the cycle. □

Computation

The motorcycle graph plays an important role for the computation of the straight skeleton. For the algorithm that will be presented in Chapter 4 it is necessary to have the motorcycle graph precomputed. Therefore, a fast algorithm for the straight skeleton is only useful if the motorcycle graph can be calculated in a fast way as well.

A basic approach for computing the motorcycle graph involves the usage of all potential crashes of the motorcycle edges. There are $\mathcal{O}(n)$ motorcycle edges which results in $\mathcal{O}(n^2)$ intersections of these edges. These intersections can then be sorted in $\mathcal{O}(n^2 \log(n))$ time by the time they occur and the actual crashes can then be calculated in chronological order. Overall this results in a $\mathcal{O}(n^2 \log(n))$ algorithm for computing the motorcycle graph of a polygon.

However, this is rather slow and it is desirable to have a subquadratic algorithm. Fortunately, it turns out that this is possible. This will be discussed in the next chapter.

3 Related Work

This chapter reviews related work and presents different approaches for computing the motorcycle graph and the straight skeleton of polygons.

3.1 Motorcycle Graphs and Straight Skeletons

Vigneron and Cheng (2007) [8] presented a new algorithm for computing the motorcycle graph that runs in $\mathcal{O}(r\sqrt{r}\log(r))$ time where r is the number of motorcycles.

Their approach is similar to the basic algorithm with $\mathcal{O}(n^2\log(n))$. While they also keep track of the chronological order they do not just calculate those straightforwardly. They use additional events and a partition of the plane. Then the algorithm runs the basic algorithm simultaneously in different regions and creates events based on motorcycles entering different regions.

They also presented a randomized algorithm which reduces the computation of the straight skeleton to a motorcycle graph in $\mathcal{O}(n\sqrt{h+1}\log^2(n))$ expected time where n is the number of vertices and h is the number of holes. Combining these with the computation of the motorcycle graph yields an algorithm for the computation of the straight skeleton in $\mathcal{O}(n\sqrt{h+1}\log^2(n)+r\sqrt{r}logr)$ expected time. For simple polygons this results in an algorithm that runs in $\mathcal{O}(n\sqrt{n}\log^2(n))$ expected time.

However, they did not provide experimental results or an implementation of their algorithm.

3.2 A Faster Algorithm for Computing Motorcycle Graphs and Straight Skeletons

Vigneron and Yan [9] presented a better algorithm in 2014 and managed to compute the motorcycle graph in $\mathcal{O}(n^{\frac{4}{3}+\varepsilon})$ time.

They use a tentative track for each motorcycle edge which can be longer than the resulting motorcycle edge in the end. At the beginning the tentative tracks are empty and then one tries to extend them one by one. This results in possible intersections and events that can be used to calculate the actual motorcycle edge step by step and not necessarily in chronological order.

In 2014 Cheng, Mencel and Vigneron [10] presented a new algorithm for computing the straight skeleton of a polygon. They also presented a deterministic algorithm that reduces the computation of the straight skeleton to the computation of the motorcycle graph in $\mathcal{O}(n\log(n)\log(r))$ time. The algorithm for comput-

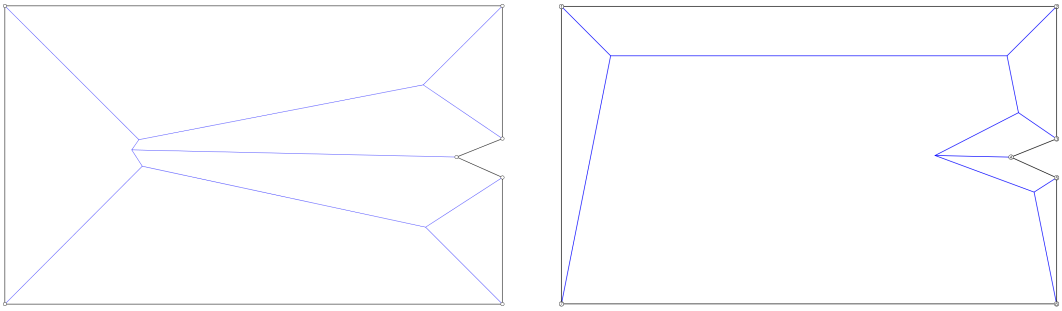
ing the straight skeleton runs in $\mathcal{O}(n \log(n) \log(r) + r^{\frac{4}{3}+\varepsilon})$ time for non-degenerate polygons.

3.3 Weighted Straight Skeletons

Aichholzer and Aurenhammer (1996) [3] introduced the straight skeleton for general planar straight line graphs.

This resulted in the concept of weighted straight skeletons. For the normal unweighted variant of the straight skeleton edges have the same speed during the shrinking process. The straight skeleton only consists of line segments of bisectors.

The weighted straight skeleton however uses weights for each edge. Each weight indicates a different speed for the corresponding line during the shrinking process. Weights are positive values and higher weights indicate a higher speed. The normal straight skeleton can be interpreted as a weighted straight skeleton with uniform weights.



(a) Unweighted **(b)** Weighted

Figure 3.1: Straight Skeleton with different weights for the bottom edge

The weighted straight skeleton does not have the same properties as the unweighted version. Its faces are, in general, not monotone. This can be see in Figure 3.1b where the face of the bottom edge is not monotone.

3.4 Straight Skeleton in Space

Straight skeletons in three dimensions were first discussed by Demaine et al. (2005) [24] and Barequet, Eppstein, Goodrich and Vaxman (2008)[25].

The concept of straight skeletons naturally transfers to polyhedra in three dimensions. In addition to vertices and edges, a polyhedron has facets and a straight skeleton results from a shrinking process where the facets move in a self-parallel manner.

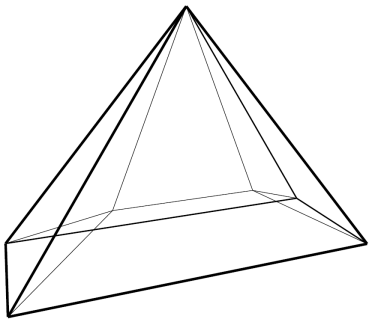


Figure 3.2: 3-dimensional straight skeleton (Walzl, 2015) [4]

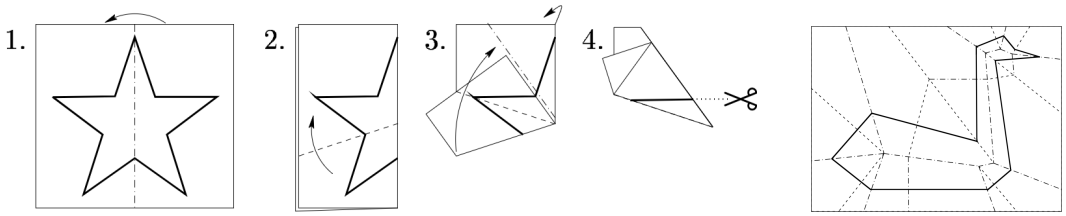
In two dimensions a vertex is connected to exactly two edges if the polygon is closed. In three dimensions the facets are moved and a vertex can be adjacent to arbitrarily many facets. This results in vertices being split during the shrinking process. An algorithm for calculating the straight skeleton in 3D is presented by Walzl (2015) [4] and more work about three dimensional straight skeletons was done by Aurenhammer and Walzl (2016) [11], including a rigorous definition and a discussion of its various constructing events.

3.5 Applications

The straight skeleton has multiple applications in computer science.

For example straight skeletons are used in computer graphics for modeling. Laycock and Day [26] show a modified usage of the straight skeleton to generate roof models. Kelly and Wonka [27] present an interactive procedural modeling system for the exterior of architectural models.

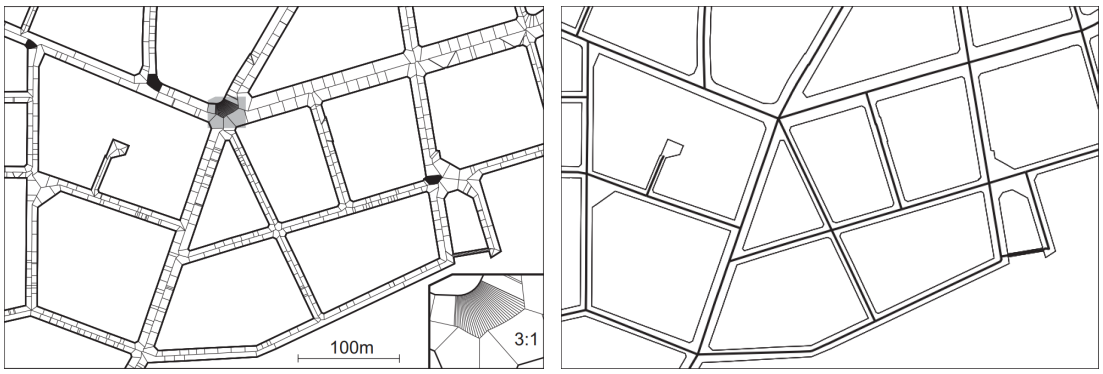
Demaine et al [23] presented an algorithm to fold a piece of paper in such a way that one single straight cut results in any given polygon for the unfolded piece of paper. This process is based on the straight skeleton to find such a fold.



(a) Creating a five-pointed star **(b)** Crease pattern

Figure 3.3: Fold-and-cut of different structures (Input from [23])

There are also applications in other areas such as geographic information science. As a skeletal structure the straight skeleton can be used as a simplified representation of objects. Haunert and Sester [28] used the straight skeleton for the collapse of small areas and finding the centerlines of roads.



(a) Straight skeleton of input data **(b)** Resulting centerlines

Figure 3.4: Finding road centerlines (Input from [28])

4 Algorithm

This chapter presents a new algorithm [1] to compute the straight skeleton $S(P)$ of a polygon P. At first, the basic idea of the algorithm is introduced and afterwards details are discussed.

The basic idea of the algorithm is to use the motorcycle graph and the resulting cells for a divide-and-conquer algorithm. The motorcycle graph partitions the polygon into convex cells, which in turn induce convex subpolygons. The straight skeleton of these subpolygons coincides with their medial axis because they are convex. The medial axis of simple polygons can be computed in linear time and therefore the straight skeleton of these subpolygons can be computed in linear time.

For the divide-and-conquer algorithm it is important to know how to divide the polygon and merge the resulting straight skeletons. This process is not straightforward because the motorcycle cells can get larger during the merge step (Figure 4.15). These details will be explained in the following subsections.

4.1 Base Algorithm

Let P be a polygon with clockwise orientation and $\mathcal{M}(P)$ the motorcycle graph of P. Let v_r be a reflex vertex and m be the outgoing motorcycle edge of v_r. Let $D_m = \{m' | m \succ m'\}$ be the set of all motorcycle edges m' that crash into m. Furthermore, let P_L and P_R be the two polygons induced by the two motorcycle cells that contain v_r. P_L corresponds to the region containing the outgoing edge of v_r and P_R corresponds to the region containing the incoming edge of v_r.

The merge algorithm is defined recursively: If D_m is empty, merge the straight skeletons of P_L and P_R otherwise compute the straight skeleton along motorcycle edges in D_m first.

In this section it is explained how the basic idea works on polygons with only one reflex vertex (Figure 4.1). Afterwards, the general case will be explained.

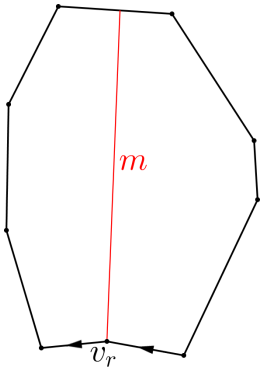
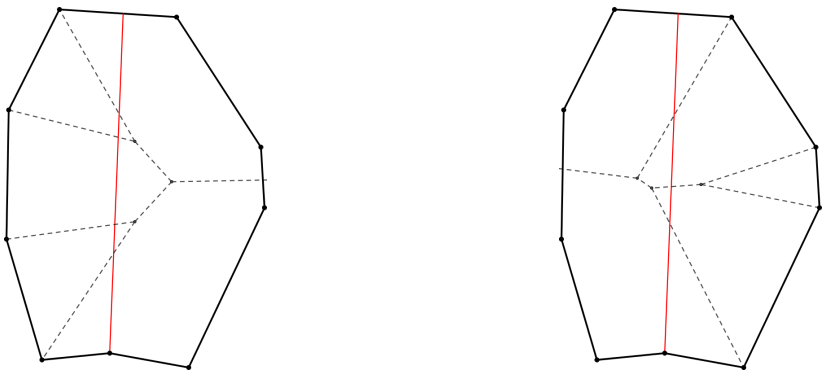


Figure 4.1: Example polygon P

It is assumed that the straight skeletons of P_L and P_R are already computed. A new polygon T is created by joining both polygons P_L and P_R at the reflex vertex r. Both polygons P_L and P_R can be open and therefore T can also be open. If the motorcycle edge m does not hit the boundary of T, it is extended until it hits the boundary of T or extends to infinity if the polygon is open and the motorcycle edge never crosses the boundary of the polygon again.



(a) $S(P_L)$ **(b)** $S(P_R)$

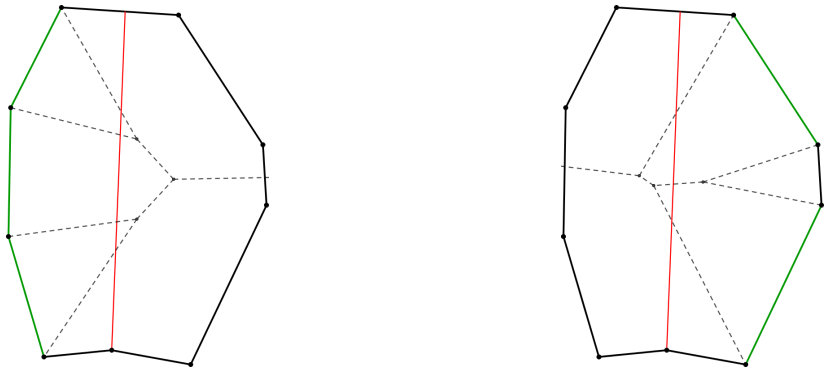
Figure 4.2: Straight skeletons of motorcycle cells

4.1.1 Overlapping Edges

Overlapping edges are the edges of the polygon that intersect the motorcycle edge m and can have an influence [1] on the other side in the final straight skeleton of P. Not all edges whose motorcycle cells overlap into the other side are relevant for the straight skeleton of the other polygon. This includes the adjacent edges of the reflex vertex v_r as well as the edge that m crashes into (This edge can be shared by both polygons). If the edge is adjacent to v_r or already part of the polygon on

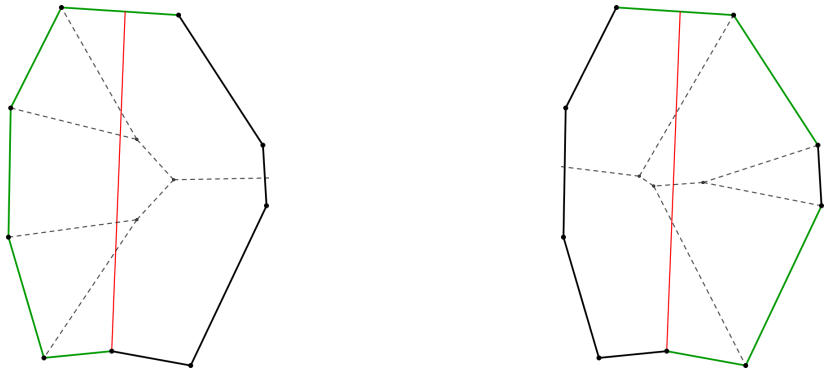
the other side then these edges are excluded from the relevant overlapping edges because they have no further influence on the other side. The adjacent edges have no influence because the arc supported by m separates the regions of the adjacent edges.

Definition 4.1 (Overlapping Edges)**.** *Let O_L (O_R) be the set of all relevant edges of P_L (P_R) whose faces intersect m.*



(a) O_L (green) **(b)** O_R (green)

Figure 4.3: Relevant overlapping edges (green)



(a) O_L with irrelevant edges **(b)** O_R with irrelevant edges

Figure 4.4: All overlapping edges (green)

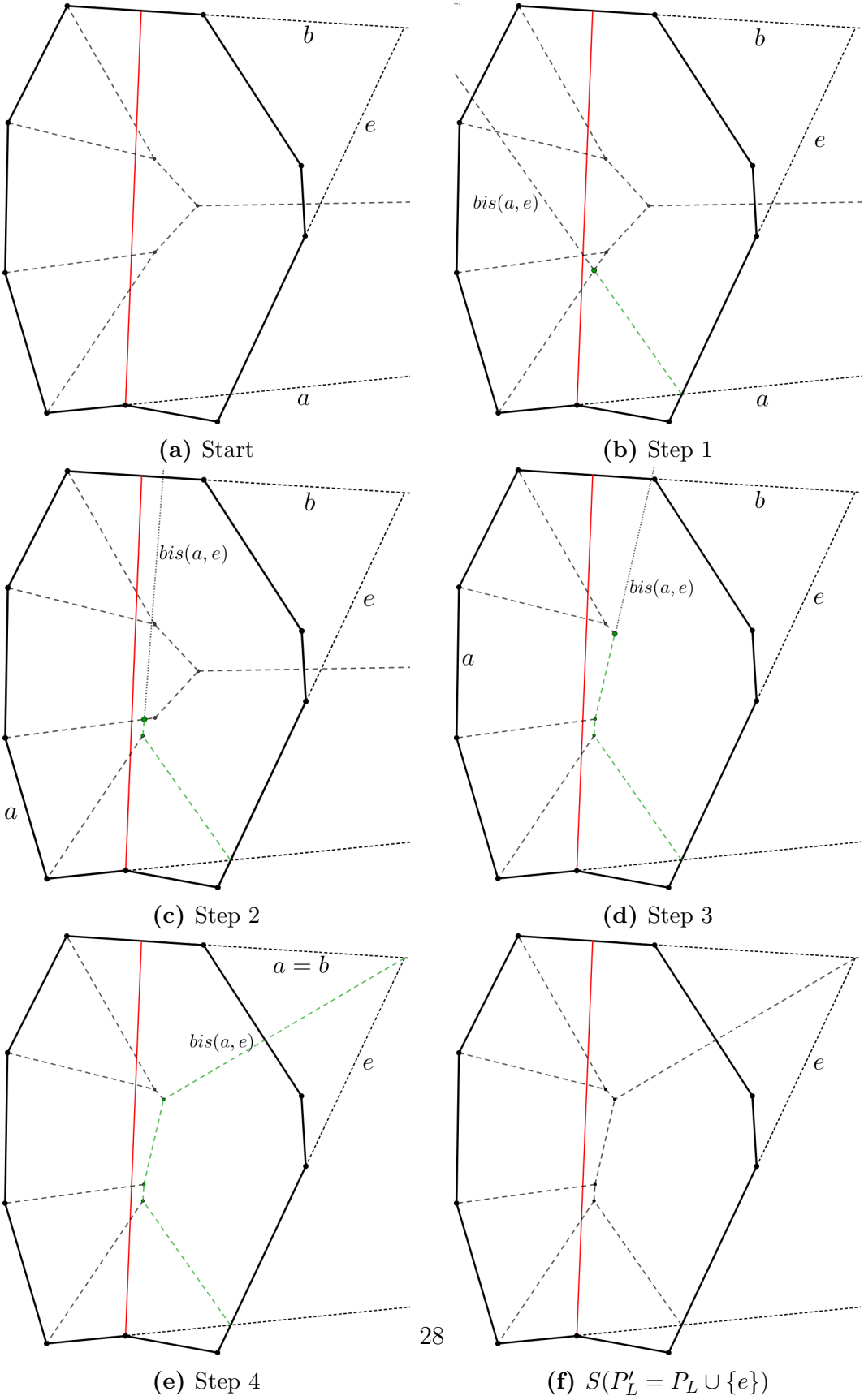
Remark. *The overlapping edges are not necessarily connected (Figure 4.3) but they form a convex polygon [1].*

4.1.2 Edge Insertion

The edges of O_L (O_R) are now inserted in random order into P_R (P_L) to update $S(P_R)$ ($S(P_L)$). This is an adaption of Chew's algorithm [12] for computing the medial axis of a convex polygon. In the following it is described how to insert an edge from O_R into the straight skeleton of P_L. Inserting edges from O_L into $S(P_R)$ works analogously.

Let $e \in O_R$. Let a and b be the edges of P_L that are adjacent to e in $P' = P_L \cup \{e\}$ (Figure 4.5a). The edge e intersects the face of a because a is adjacent to e in P'. In the next step $bis(a, e)$ needs to be calculated and the arc s_a of $S(P_L)$ on the boundary of the face of a that intersects this bisector needs to be identified. This intersection point marks the end point of $bis(a, e)$ and s_a. Besides the update of these two arcs the region needs to be updated too: Obsolete edges (dominated by $f(e)$) are removed from the straight skeleton.

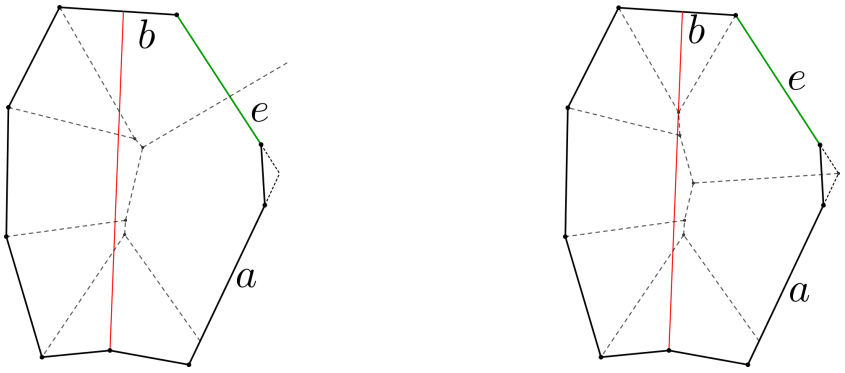
Furthermore, the intersected arc s_a is supported by a (as it is part of the region of a) and another edge x. The algorithm continues by updating $a \leftarrow x$ and doing the same update step again. This procedure continues until a matches b and the last intersection is simply connected to the intersection of b and e. An example of the insertion of the first edge is shown in Figure 4.5.



(a) Start **(b)** Step 1 **(c)** Step 2 **(d)** Step 3 **(e)** Step 4 **(f)** $S(P'_L = P_L \cup \{e\})$

Figure 4.5: Insertion of e into $S(P_L)$

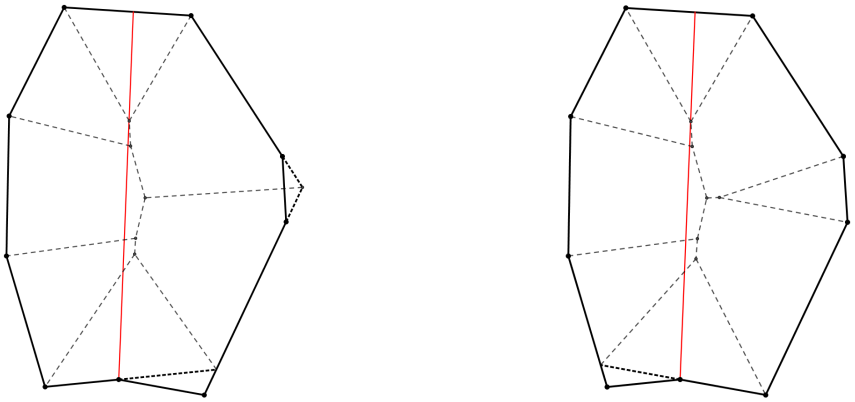
Afterwards the polygon is updated $P_L \leftarrow P_L \cup \{e\}$ and the edges from O_R are inserted until there are none left. This results in the straight skeleton of the left side.



(a) Start **(b)** $S(P_L'' = P_L' \cup \{e\})$

Figure 4.6: Adding the final edge to the left side (green)

The same process is used to calculate the straight skeleton of the other side. Now that the straight skeletons of both sides are finished they need to be merged.



(a) $S(P_L)$ **(b)** $S(P_R)$

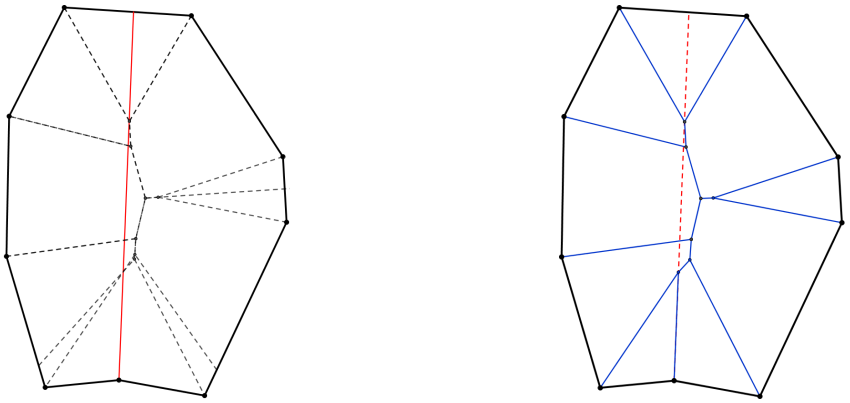
Figure 4.7: Straight skeleton of each side

4.1.3 Merge Process

Observation. *The straight skeletons $S(P_L)$ and $S(P_R)$ coincide with the final straight skeleton along their defining motorcycle edge m.*

This is because of the supported edges of the arcs of $S(P_L)$ and $S(P_R)$ that cross the motorcycle edge that also exist in P. In particular, the straight skeletons of each side coincide along the motocycle edge. Let R be the region that results from merging the left and right region of the reflex vertex. Finally, $S(R)$ is obtained by merging the straight skeletons $S(P_L)$ and $S(P_R)$ along the motorcycle edge m.

Lines that cross m and share the same intersection as a line from the other side are supported by the same edges with the exception of the intersection closest to the reflex vertex. This exception is due to the fact that the arcs that cross the motorcycle edge closest to the reflex vertex are supported by the adjacent edges of the reflex vertex. These edges are not included for the computation of the straight skeleton of the other side as they have no influence.



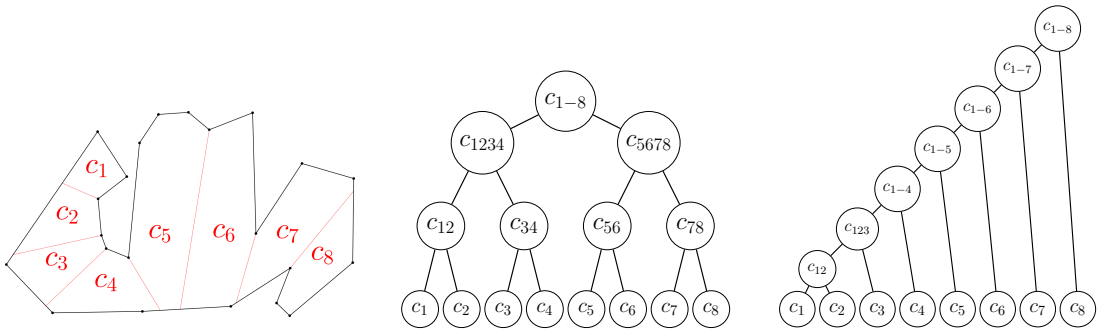
(a) Overlay of both straight skeletons **(b)** Final straight skeleton

Figure 4.8: Merge process

Definition 4.2 (Merge Tree)**.** *Let P be a polygon and $\mathcal{M}(P)$ its motorcycle graph. The merge processes induce a graph $T = (V, E)$ where V represents each region induced by $\mathcal{M}(P)$ and every new region that is created by the merge process. Every new region is connected by an edge to the regions that define it and form the set of edges E.*

Remark. *Every merge of two regions results in a new region and this results in $r - 1$ created regions during the merge process. Therefore, a merge tree contains $r + (r - 1) = 2r - 1$ vertices.*

The height of a merge tree depends on the merge order of the motorcycle edges. In general, various different merge trees are possible (Figure 4.9).



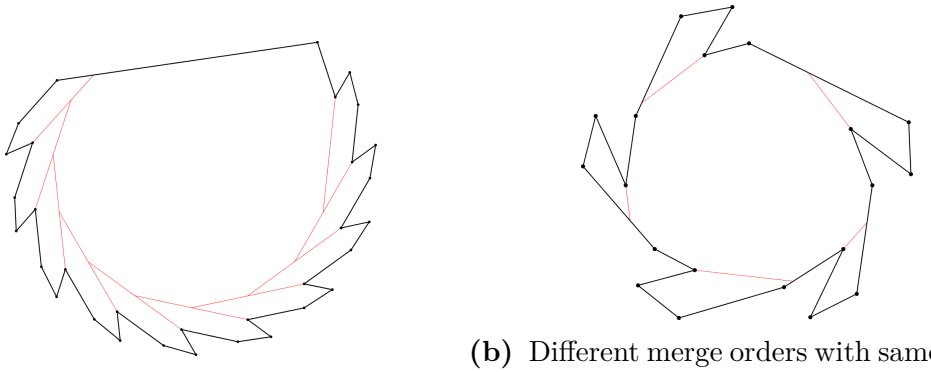
(a) Polygon with cells **(b)** Balanced **(c)** Linear height

Figure 4.9: Polygon with different possible merge trees

Theorem 4.3 (Aurenhammer and Steinkogler, 2018 [1])**.** *The straight skeleton of a polygon P can be computed in $\mathcal{O}(d\,n\log(n))$ expected time using this algorithm where d is the height of the merge tree.*

Remark. *The optimal height of a merge tree is $\mathcal{O}(\log(n))$ if the tree is balanced and $\mathcal{O}(n)$ in the worst case. This results in an expected running time of $\mathcal{O}(n\log^2(n))$ in the best case, which competes with the best algorithms currently known.*

Certain motorcycle graphs dictate a structure of the merge tree because regions cannot be combined arbitrarily (Figure 4.11). Some may force a balanced tree. However, these polygons have a good expected running time in the first place. Therefore, polygons with merge trees that have a worse structure are of interest. Polygons with such structure might have only one possible merge order which results in only one possible outcome of the merge tree. Other polygons might have multiple possible merge orders but all result in the same bad structure of the merge tree. Examples of such polygons are shown in Figure 4.10.



(a) Only one merge order **(b)** Different merge orders with same merge tree

Figure 4.10: Polygons with merge trees of linear height

4.2 Problems & Difficulties

The following sections cover certain problems that occur for the base algorithm and present solutions for it.

4.2.1 Merge Order

It is sometimes possible to merge along motorcycle edge that dominate unmerged motorcycle edges. However, this is not the case in general. There are several problems that occur when such a motorcycle edge is used during the algorithm:

1. Reflex vertices are overlapping into other regions which creates non-convex polygons that overlap the current motorcycle edge.
2. Overlapping edges can cut off huge parts of the other region polygon and/or create non-convex polygons.

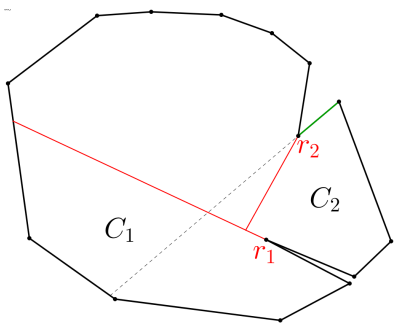
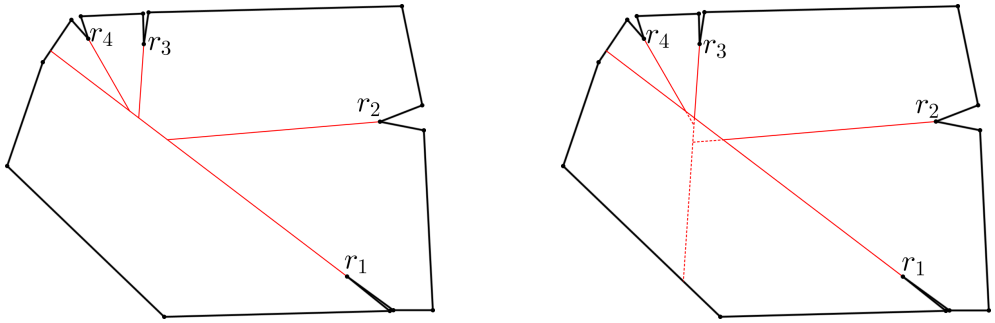


Figure 4.11: Merge along dominant motorcycle edge (Using r_1)

As long as there are no cycles there will always be at least one motorcycle edge that does not dominate any other unmerged motorcycle edge. In these cases it will never be necessary to merge along such motorcycle edges.

However, a similar problem arises when motorcycle edges are merged that dominate no other motorcycle edge. This can be seen in Figure 4.12. It is possible to start with the outgoing motorcycle edges of the reflex vertices r_2, r_3 and r_4 because they do not dominate any other active motorcycle edge. However, merging along the outgoing motorcycle edge m_{r_3} of the reflex vertex r_3 is not a good idea because this essentially results in a merging process with motorcycle edges that dominate unmerged motorcycle edges as described above.



(a) Motorcycle graph **(b)** Extended motorcycle graph

Figure 4.12: Lost information

The merge process is shown in Figure 4.13. Merging along m_{r_3} works fine and results in the correct straight skeleton for the subpolygon. However, the faces corresponding to the edges adjacent to the reflex vertex r_3 overlap m_{r_4}. In the final straight skeleton this overlap should not be there anymore.

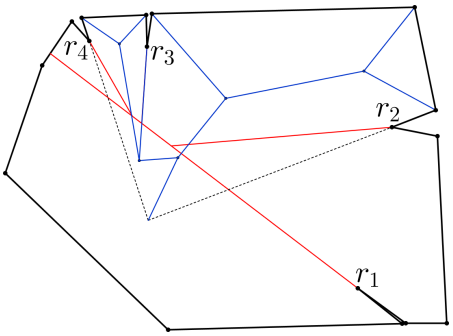
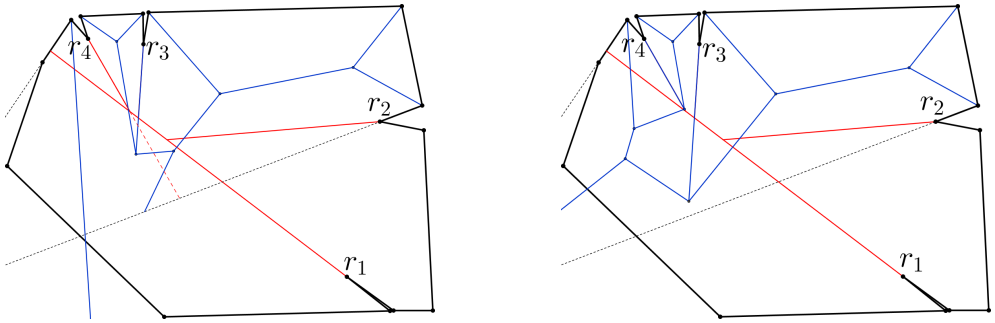


Figure 4.13: Merge along m_{r_3}

Trying to merge the straight skeletons along m_{r_4} (Figure 4.14a) results in two problems

1. Right side: The overlapping faces (from the left side) of the adjacent edges of r_3 result in a non-convex overlapping polygon and the merge process is either undefined or results in a wrong outcome for the right side.

2. Left side: There is no overlapping edge from the right side. This means that the left side is not updated correctly. The correct straight skeleton for this subpolygon can be seen in Figure 4.14b. One adjacent edge of r_4 had an influence on the straight skeleton arc corresponding to m_{r_3} and cuts it off. By merging along m_{r_4} this influence is removed and this arc needs to be extended now.



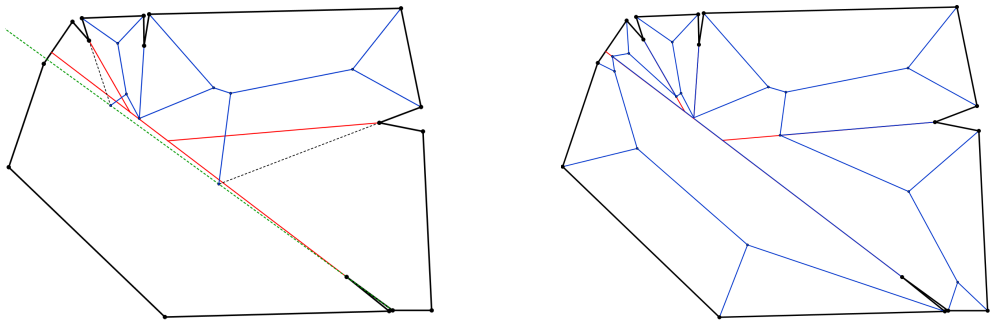
(a) Merge along m_{r_3} **(b)** Merge along m_{r_4} then m_{r_3}

Figure 4.14: Comparison between different merge orders

Some motorcycle edges remove important information about the motorcycle graph of subpolygons (Figure 4.12b). Reconstructing the motorcycle graph from subpolygons of P by using $\mathcal{M}(P)$ or simply computing a new motorcycle graph to get a correct merge order of subpolygons is not efficient.

Solution

The key issue for these problems is the overlap of edges over motorcycle edges that should not happen in the first place. To solve these issues the adjacent edge of the dominant motorcycle edge is added to both sides (if it is not already included) as an overlapping edge. For the polygon in Figure 4.12a this corresponds to the right edge of r_1.



(a) Merge with overlapping dominant edge **(b)** Final straight skeleton

Figure 4.15: Solution of the merge order problem

4.2.2 Single Edge Regions

One of two regions contains only one edge which means that there is no straight skeleton to begin with.

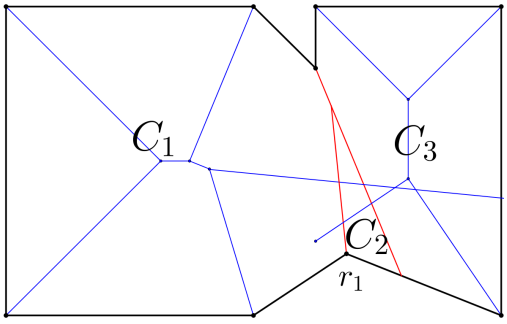
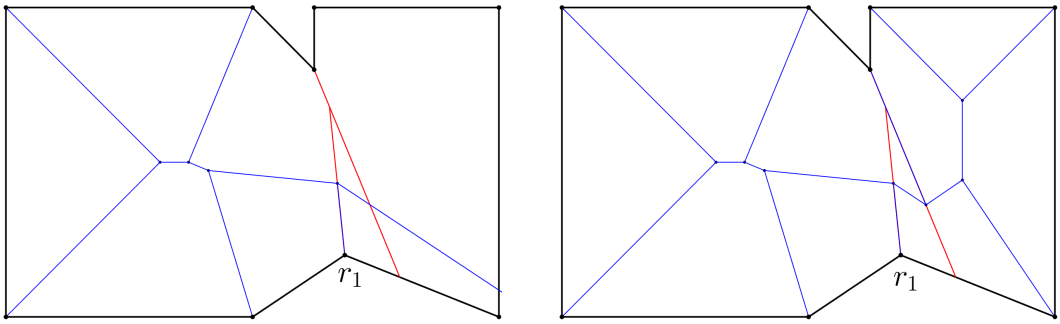


Figure 4.16: Region with only one polygon edge (C_2)

P_R is the polygon with only one edge. There is no edge that needs to be inserted to $S(P_L)$ because the only edge in P_R is adjacent to the reflex vertex and has no influence on the straight skeleton of the left side.

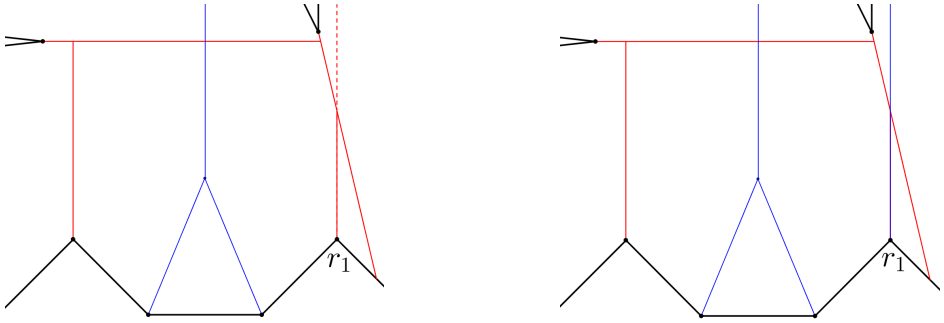
On the other side, all edges that overlap the motorcycle edge of r_1 are combined with the one edge of P_R and induce a convex polygon. It is easy to compute the straight skeleton of this polygon. Afterwards the straight skeletons of both sides can be merged along the motorcycle edge.



(a) $S(C_1 \cup C_2)$ **(b)** $S(P)$

Figure 4.17: Adding the final edge to the left side (green)

However, it is also possible that there is no arc of $S(C_1)$ that overlaps the motorcycle edge in the first place. This can be seen in Figure 4.18a. It turns out that this is easier to solve because there is nothing that needs to be added on each side. Only a ray supported by the motorcycle edge needs to be added for the final straight skeleton of these two polygons.



(a) No overlapping edges **(b)** Solution

Figure 4.18: Disconnected straight skeleton

This also works if both polygons consist of only one edge. It is also possible that there are arbitrarily many consecutive regions consisting of only one edge. A relevant polygon is shown in Figure 4.19

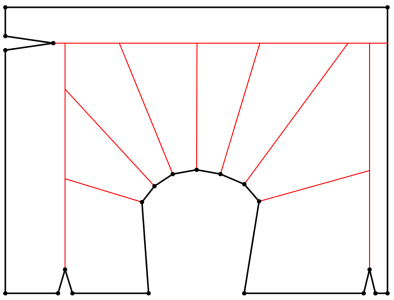


Figure 4.19: Polygon with multiple neighbored one-edged regions

Another approach is adding the adjacent edge of the dominant motorcycle edge to the single edge region which creates a convex polygon. This gets rid of single edge regions. Then the straight skeleton (in this case only a single arc) of this convex polygon can be computed and the merge process can be done normally.

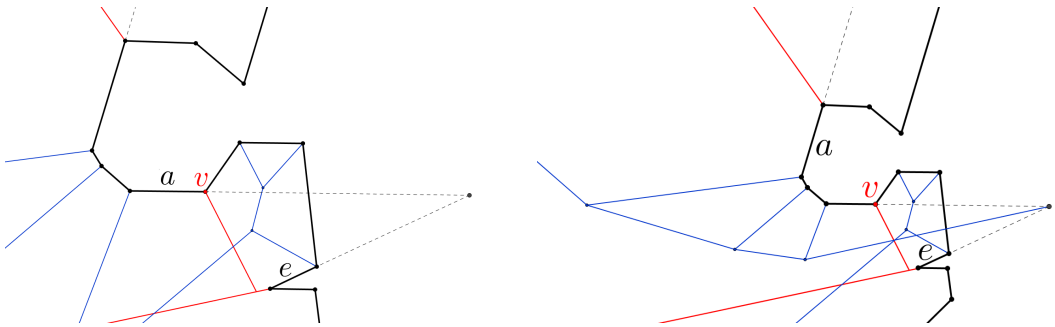
4.2.3 Open Polygons

The algorithm starts with $bis(a, e)$ and ends if it reaches the region of b. However, there are two problematic cases for the definitions of a and b:

1. The inserted edge e has only **one** intersection with the region polygon.
2. The inserted edge e has **no** intersection with the region polygon.

One Intersection

This can happen if one of the polygons is open. An example is shown in Figure 4.20a: The edge e intersects the polygon only in one point (Intersection with a). In this case the algorithm starts as usual with $bis(a, e)$ and updates a as usual. It stops when $bis(a, e)$ does not hit the boundary of a face anymore which means that $bis(a, e)$ is extended to infinity.



(a) Only one intersection **(b)** Solution

Figure 4.20: Special case with only one intersection

It is not always the case that the last updated edge a coincides with the open edge of the polygon on the other side (Figure 4.22b).

No Intersection

This is essentially the same problem as described in the previous section and can happen for open polygons. Inserting an edge e might result in no intersection with the polygon on the other side at all (Figure 4.21). Therefore, the edges a and b are not defined as described in the base algorithm.

Let P be the region polygon of one side that results in such an open polygon. The resulting merged polygon $P \cup \{e\}$ is open around the motorcycle edge. This means that the edge of P that coincides with one of the adjacent edges of the reflex vertex is needed for the first step of the computation. This edge is used as the edge a for further computations.

The edge b does not need to be defined yet as seen in the previous section. The insertion procedure simply continues again until $bis(a, e)$ no longer intersects the boundary of a face.

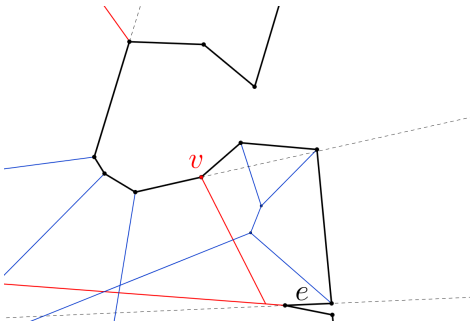
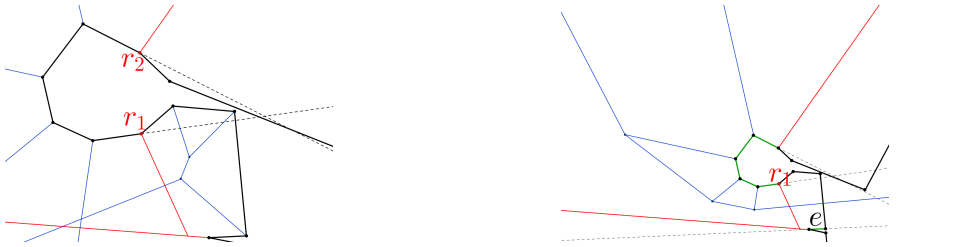


Figure 4.21: No intersection

4.2.4 Induced Polygons with Crossings

The original polygon P is simple. However, this is not true for the induced polygons during the merge process.

Merging along the reflex vertex r_1 (Figure 4.22a) results in a self-intersection of the polygon to its right side.



(a) Self-intersecting polygon **(b)** Resulting polygon (green edges)

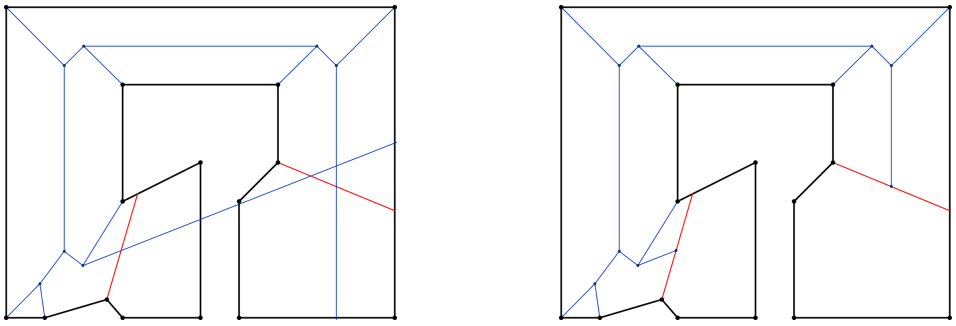
Figure 4.22: Induced polygon with crossing

This also implies that the straight skeletons during the merge process do not need to be crossing-free.

The overlapping edges can be affected by this, because there can be lines from the straight skeleton that cross the current motorcycle edge without having any influence on the merged skeleton (Figure 4.23a). This means that they have to be excluded from the definition of the overlapping edges.

To avoid problems during the selection of overlapping regions, the lines of the straight skeleton can be trimmed with respect to the region polygon and its motorcycle edges (Figure 4.23b).

These crossings do not occur locally which means that it is possible to ignore these edges by looking at the cells around the motorcycle edge.

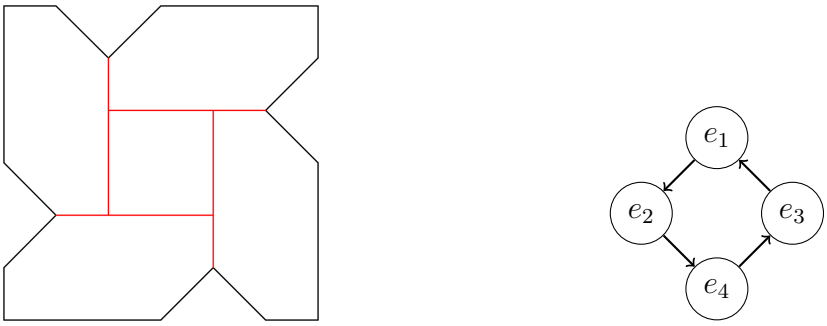


(a) Self-intersecting straight skeleton **(b)** Bounded by motorcycle edges

Figure 4.23: Self-intersecting straight skeleton

4.3 Cycles

Cycles cannot be resolved normally because the dominance graph has a cycle in it.



(a) Motorcycle graph **(b)** Dominance graph

Figure 4.24: Cycle in a motorcycle graph

Lemma 4.4. *A cycle in the underlying undirected graph of the dominance graph can only correspond to a cycle in the motorcycle graph.*

Proof. Let $C = \{c_1, \ldots, c_k\}$ be a cycle in the dominance graph. If there is a permutation of C with $c_{\pi(1)} \succ c_{\pi(2)} \succ \ldots \succ c_{\pi(k-1)} \succ c_{\pi(k)} \succ c_{\pi(1)}$ (all edges have the same orientation) then C corresponds to a cycle in the motorcycle graph by definition. Otherwise, there exist vertices c_i, c_j, c_k with $c_j \succ c_i$ and $c_k \succ c_i$ which is not possible because a line in the motorcycle graph can only crash into exactly one other line. □

Lemma 4.5. *There is at most one cycle per component in the dominance graph.*

Proof. Assume there are at least two cycles C_1 and C_2 in one component.

There is no undirected path from vertices in C_1 to another vertex in C_1 without using vertices from C_1 (4.4). Let $c, d_2 \in C_1$. There can only be one incoming edge per vertex and c and d already have one incoming edge from a vertex in C. Therefore, a path from c to d has an outgoing edge from c in the beginning and an outgoing edge from d at the end. Those two edges point in different directions which implies that there is at least one vertex in this path with two incoming edges and thus contradicting the fact that there can only be on incoming edge.

C_1 and C_2 are in the same component and therefore there exists a path from $c_1 \in C_1$ to $c_2 \in C_2$. This path does not include any other vertex from C_1 or C_2. The path from c_1 to c_2 has outgoing edges from c_1 and c_2 with opposite directions at each end and that results again in a vertex with two incoming edges. This contradicts the assumption that there are at least two cycles in one component. □

Before resolving the cycle all other motorcycle edges in the component need to be resolved. This is possible due to Lemma 4.5 and the cycle will be the only thing that is left to merge in one component of the dominance graph.

In the following it is shown w.l.o.g. how to resolve a motorcycle graph that only consists of a cycle.

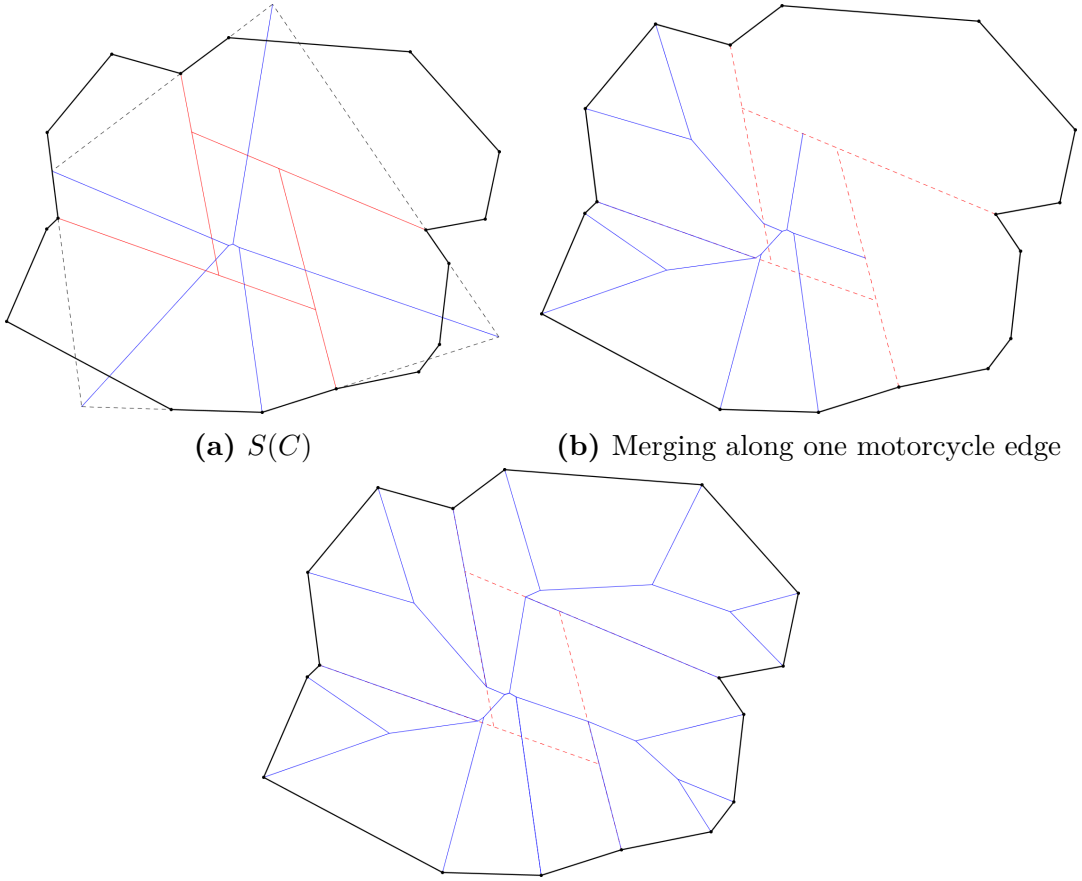
Merging along a motorcycle edge in the absence of cycles results in adding all relevant edges from both sides with a possible extra edge from the dominant motorcycle edge. However, simply merging together two regions along one motorcycle edge that is part of a cycle like this results in multiple problems. A motorcycle edge like this is not shared fully by two adjacent regions which effectively results in the same problem as merging along a motorcycle edge that dominates unresolved motorcycle edges. In the case of cycle-free motorcycle graphs this could be avoided by merging the dominated motorcycle edges first. This is not possible for a cycle since there is no such line.

The inner motorcycle cell is used to solve this problem. Let C be this motorcycle cell and m be the motorcycle edge of the cycle that is currently used in the merge process. m is covered by the two adjacent regions of m that are left and right of its reflex vertex and C. Every edge of the polygon that will have a face that intersects m has to either come from the left or right region or has to sweep over C.

The straight skeleton $S(C)$ of C is calculated by using all edges of the polygon whose regions overlap the cycle (Figure 4.25a). These edges form a convex polygon.

The overlapping edges (with respect to the common part of m) and the edges from C that overlap from the cycle into the outer region are inserted into the other outer region normally (Figure 4.25b). Resolving the cycle like this results in the final straight skeleton $S(P)$ (Figure 4.25c).

This process can be done in $\mathcal{O}(l \log(l))$ expected time where l is the total number of overlapping edges [1]. Merging along motorcycle edges step by step results in a dominance graph with linear height. However, this can be improved by merging pairs of adjacent regions together first and resolving them like a binary tree structure. This can be used to achieve an overall runtime of $\mathcal{O}(k \log^2(k))$ where k is the size of the cycle.



(a) $S(C)$ **(b)** Merging along one motorcycle edge

(c) $S(P)$

Figure 4.25: Merging cycles

4.3.1 Properties of intermediate skeletons

The underlying structure of the straight skeleton of a simple polygon is a plane tree. This chapter showed that the straight skeletons created during the algorithm do not necessarily have this property. The straight skeletons can be disconnected (Figure 4.19) and/or have crossings (Figure 4.23b).

The straight skeletons created in each merge step equal the actual straight skeletons of the region polygons if the motorcycle graph does not contain a cycle. For cycles this is true if the whole cycle is resolved (Figure 4.25c). In general, the straight skeleton resulting from merging along a single motorcycle edge of the cycle only coincides with the actual straight skeleton on the left and right motorcycle cell and the center region (Figure 4.25b). An example of a straight skeleton that does not coincide with the actual straight skeleton outside of these cells is shown in Figure 4.26.

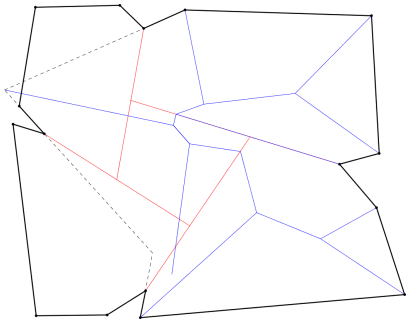


Figure 4.26: Calculated straight skeleton and the region polygon

5 Implementation

This chapter will discuss the implementation of the algorithm presented in Chapter 4. The program has been implemented with Java (JSE 7) using Piccolo2D for the visualization.

Usage of the program and compatibility to GeoGebra will be discussed briefly. Furthermore, used data structures and explanations for key parts of the implementation will be discussed as well.

5.1 Program

The interface of the program is inspired by GeoGebra.

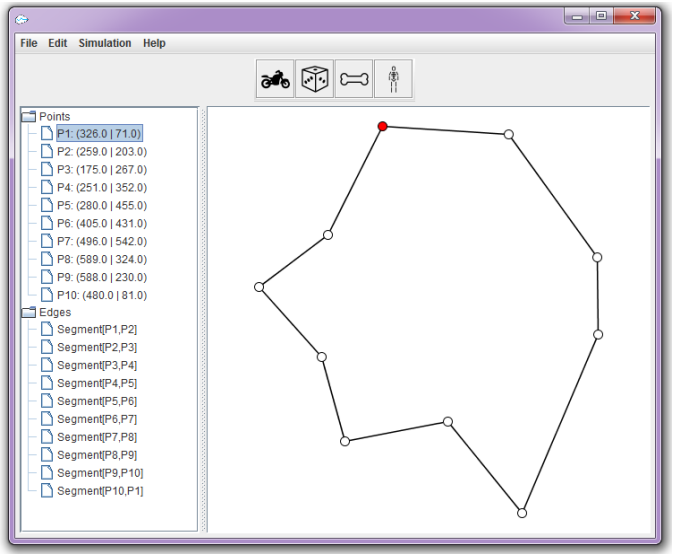


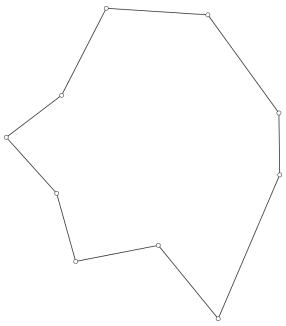
Figure 5.1: User interface

5.1.1 Input

The program needs a closed simple polygon as input. This can be either added manually using the interface or by importing a .dat file. The file contains a list of vertices where each vertex is connected to the one below it. The polygon is closed if the last vertex in the list is equal to the first one and open otherwise. An open polygon needs to be closed manually because the program cannot compute the straight skeleton of an open polygon for multiple reasons. The main reasons are that it is sometimes hard to decide what should be the inside of the polygon and the polygon can be self-intersecting by simply closing it.

```
326.0 71.0
259.0 203.0
175.0 267.0
251.0 352.0
280.0 455.0
405.0 431.0
496.0 542.0
589.0 324.0
588.0 230.0
480.0 81.0
326.0 71.0
```

(a) .dat file



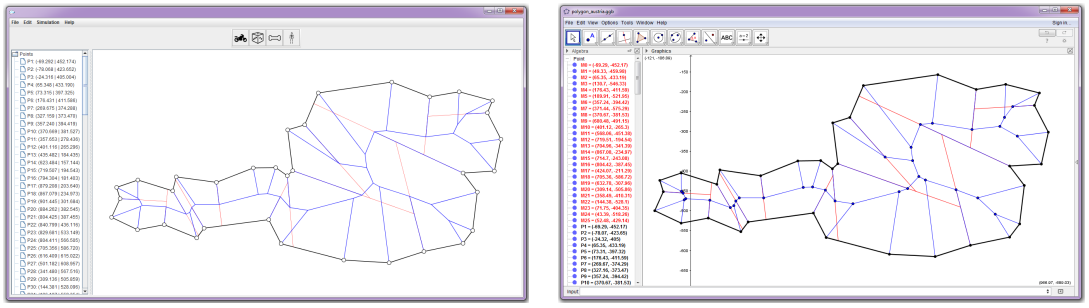
(b) Polygon

5.1.2 Output

The program is capable of calculating a few things related to the straight skeleton. Besides the calculation of the straight skeleton and the motorcycle graph, the medial axis of each convex region polygon can be calculated (The straight skeleton coincides with the medial axis for convex polygons). This can be used to merge along reflex vertices by hand.

Furthermore, it is possible to run a folder of polygons and analyze parameters of the straight skeleton for each of them to obtain experimental results. This will be covered in Chapter 6.

There is also the possibility to export the straight skeleton to a .txt file due to the fact that GeoGebra is a way more powerful tool and general gemeotric manipulations can be performed there instead. After a successful export there will be a file containing two commands for GeoGebra: One to import all points and line segments and another one to format them.



(a) Original polygon

(b) Exported to GeoGebra

Figure 5.3: Polygon output

5.2 Data Structures

5.2.1 Polygons

Polygons are represented by an ordered list of vertices and edges. Each vertex has an incoming edge and an outgoing edge.

All edges are directed and the polygon is represented by a directed cycle. Furthermore, each edge stores information about its start and end vertex.

5.2.2 Motorcycle Graphs

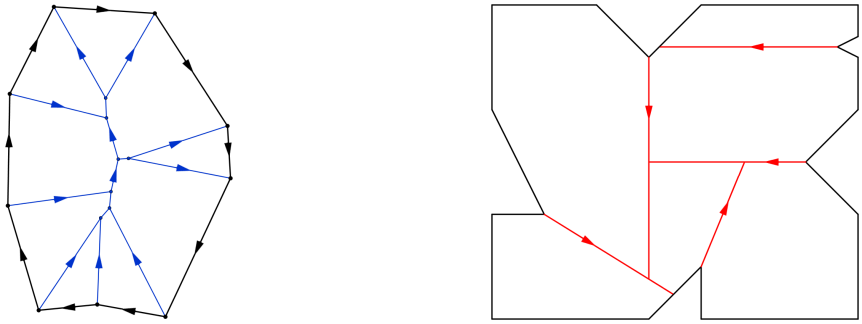
The motorcycle graph is represented by an arrangement of directed edges. Each motorcycle edge l has the following information:

- The reflex vertex defining the start of the motorcycle edge
- The dominant motorcycle track that l crashes into
- A set of motorcycle edges that crash into l

5.2.3 Straight Skeletons

The tree structure of the straight skeleton is represented by an unordered list of vertices and edges. Each edge is directed and has a start and an end vertex.

Each vertex of the straight skeleton has a list of adjacent edges.



(a) Polygon and straight skeleton **(b)** Motorcycle graph

Figure 5.4: Structures

5.2.4 Motorcycle Cells

Motorcycle cells consist of edges of the polygon that are part of the motorcycle cell and motorcycle edges that separate it from other motorcycle cells. In addition, the region polygon induced by the edges and the reflex vertices of the motorcycle cell are stored.

5.2.5 Reflex Vertices

A reflex vertex stores the following information

- The outgoing motorcycle edge
- The adjacent edges from the original polygon
- The adjacent motorcycle cells

5.3 Details

5.3.1 Polygon Orientation

The orientation of the polygon is calculated by using the shoelace formula.

Lemma 5.1 (Shoelace formula [20] [21])**.** *Let P be a planar non-self-intersecting polygon with vertices $(x_1, y_1), \ldots, (x_n, y_n)$. The signed area A of the polygon P is given by*

$$A = \frac{1}{2}\left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \ldots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix}\right)$$

The points are in clockwise order for a normal cartesian coordinate system if the area of the polygon is negative and in counterclockwise order otherwise. However, the program uses an inverted y-axis and therefore this rule has to be reversed.

The orientation is used to decide which angle spanned by two polygon edges is inside the polygon

5.3.2 Detecting Induced Polygons

The motorcycle graph induces convex polygons. The edges for such a polygon are detected by using the motorcycle graph and is done iteratively. The algorithm runs over all polygon edges and motorcycle edges of a cell and detects regions in that way.

Start with an arbitray reflex vertex and follow its motorcycle cell by using the orientation of the polygon and the motorcycle graph. A motorcycle cell is convex so this can be used to iterate over all parts of the cell until the reflex vertex is reached again. All visited edges of the polygon are added to create the induced polygons.

Each region is connected to at least one reflex vertex so this algorithm is applied to get each incoming and outgoing region. This is not true for inner motorcycle cells created by cycles but these are irrelevant for the creation of the induced polygons in the first place.

Lemma 5.2. *Detecting the regions of P induced by $\mathcal{M}(P)$ can be done in $\mathcal{O}(n)$ time by using the presented algorithm if the dominated edges of each motorcycle edge are sorted and $\mathcal{O}(n \log(n))$ time otherwise.*

Proof. Assume all crashed edges are sorted. The proof is presented in two parts. First the number of polygon edges are counted and afterwards the number of motorcycle edges that are visited during the algorithm.

Let n be the number of vertices and r the number of reflex vertices. Each edge of the polygon is part of at least one cell and an edge is part of multiple cells if a motorcycle edge crashes into it. This results in at most $n + r$ edges on the polygon that are counted during the algorithm. The upper bound is achieved by a motorcycle graph without intersections.

Every edge of the motorcycle graph is part of at least two cells. A motorcycle edge is part of one extra region for each intersection on it. There are at most r intersections. This results in at most $2r + r$ counted line segments.

Together this results in

$$\sum_{\text{Cell } c} |c| \leq n + r + 2 \cdot r + r \leq 5 \cdot n = \mathcal{O}(n).$$

However, the dominated edges of each motorcycle edge have to be sorted first to find the next point of a region on a motorcycle edge. Each motorcycle edge is processed separately in $\mathcal{O}(r_i \log(r_i))$ time where r_i is the number of crashed edges on the motorcycle edge m_i. There are r end points, one for each motorcycle edge and an end point lies on at most two edges. Therefore, $\sum_i^r r_i = \mathcal{O}(n)$ and the points can be sorted in

$$\sum_i^r r_i \log(r_i) \leq \sum_i^r r_i \log(r) = \left(\sum_i^r r_i \right) \log(r) = \mathcal{O}(n \log(n))$$

time. □

5.3.3 Bisector Computation

Remark. *Two points A, B lie on the opposite side of a line l if and only if the segment AB and l have an intersection point.*

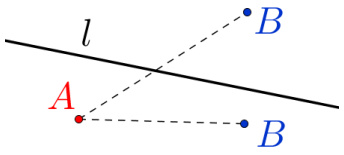


Figure 5.5: Orthogonal projection

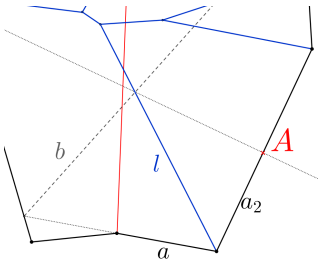
A bisector of two non-parallel lines is not uniquely defined because there are two possible bisector lines. Furthermore, bisectors are generated as rays with the intersection of its supported lines as the start vertex. This results in four possible bisectors during the calculation but only one of them will be relevant for the calculation of the straight skeleton.

Furthermore, bisectors that are added during the insertion step do not always start in the intersection point of its two supported lines. This makes it necessary to calculate the correct side where the bisector needs to start, which is either the intersection of its supported lines or a point at infinity.

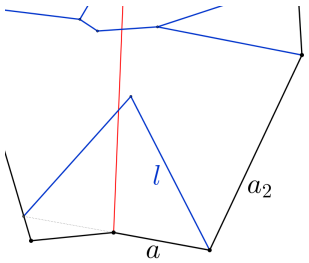
Let b be the bisector generated by the inserted edge e and an edge a of the polygon. Furthermore, let l be the line of $f(a)$ that b crashes into and a_2 be the supporting edge of l that is not a. To determine the start point of b one can use the intersection of a and e and a point A on the line created by a_2 that lies on the motorcycle cell of a.

Lemma 5.3. *The orthogonal projection of $b \cap l$ onto l_{a_2} is a point inside of the motorcycle cell of a_2 (Figure 5.6a).*

Proof. Monotonicity of the straight skeleton. □



(a) Orthogonal projection



(b) Resulting line segments

Figure 5.6: Adding first bisector

This point A can now be used to decide whether the intersection of e and a is the start vertex or not. The same thing can be done to determine the start and end points of the line l by using the bisector b in the process. The next bisector is calculated in the same way by using the line l to calculate the correct direction.

5.3.4 Edge Selection

The selection of edges a and b described in Subsection 4.1.2 with its problems is not always easily possible and is avoided in general.

Edge a is always selected as the edge connected to the reflex vertex. On the other side, there are several problems that arise for the selection of b. b is in general not the last edge of the induced polygon on one side. Furthermore, the motorcycle cell of b does not necessarily cross the motorcycle edge and is therefore not always an overlapping edge. This possibly results in the problem that a, b and the inserted edge e do not define a convex polygon. In addition, not all of these edges have to be relevant for the construction of the polygon in the first place (Figure 5.8).

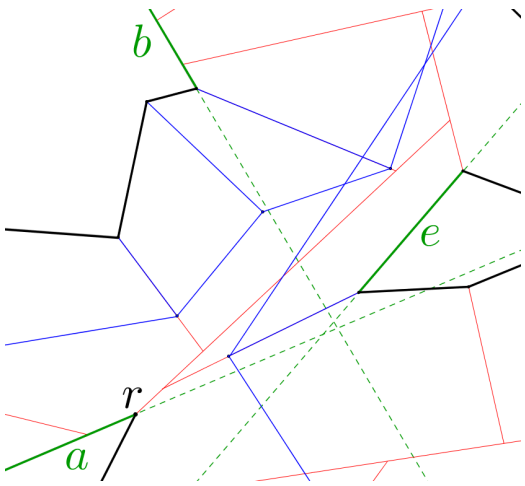


Figure 5.7: Edges a, b and e

5.3.5 Overlapping Edges

Finding edges whose faces overlap the motorcycle edge can be done by checking arcs of the straight skeleton around the motorcycle edge.

The following describes the identification of O_R. Finding edges of O_L works analogously. For a given reflex vertex r start with the motorcycle region of the right side. If there are overlapping arcs then there is at least one that is part of this region namely the one closest to the reflex vertex. The overlapping edges can be identified by iterating over the arcs of the straight skeleton of this motorcycle region until there is either an intersection with the motorcycle edge or no intersection can be found. If there is no intersection then there are no overlapping edges in the first place. Otherwise, an overlapping arc will be found and the supporting edge e which is not the edge connected to the reflex vertex will be added to O_R. However, the edge will not be added if it is already contained in P_L.

Afterwards the arcs of $f(e)$ will be iterated over and this process is repeated until there is no arc left in the region and no further intersection can be found.

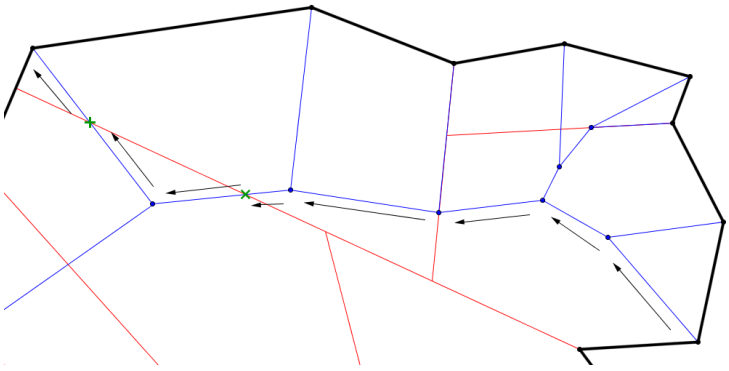


Figure 5.8: Finding overlapping edges

6 Experimental Evaluation

This chapter presents experimental results of the algorithm presented in Chapter 4 by using the program discussed in Chapter 5.

The running time of the algorithm depends on the height of the merge tree. Therefore, there is a special interest in polygons with high merge trees. Their expected running time is $\mathcal{O}(n^2 \log(n))$ in the worst case. Some of these polygons will be examined and discussed.

6.1 Parameters

Let P be a polygon of n vertices with r reflex vertices. The number of merge steps is constant for a fixed number of reflex vertices because every vertex needs to be used exactly once. For each merge step a number of parameters are of interest: The number of edges that need to be inserted into each side, the number of iterations for each inserted edge and the time the algorithm needs to finish the task. The time measures the amount of time needed for the iterations after the motorcycle graph and the medial axis of all induced polygon are calculated.

This will lead to a total number and an average number per reflex vertex for most parameters.

6.2 Tested Polygons

The main interest during the evalution of different polygons is not necessarily the structure of the polygon itself but also the structure of the motorcycle graph. These tested polygons include some special motorcycle graph structures and then also more general polygons will be discussed.

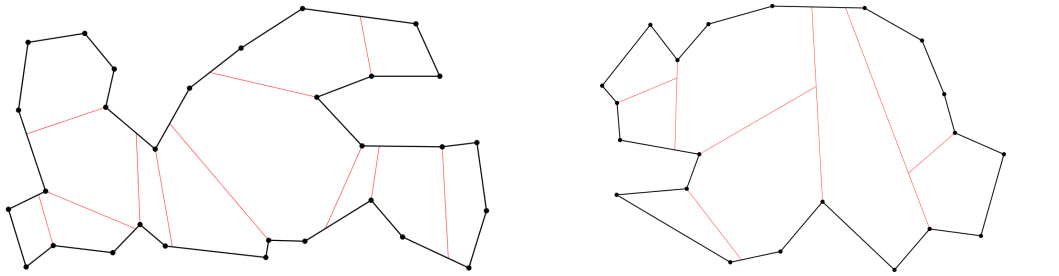
6.3 Disconnected Motorcycle Graphs

Definition 6.1 (Disconnected Motorcycle Graphs)**.** *Let $\mathcal{M}(P) = \{t_i \,|i \in \{1, \ldots, n\}$ be the motorcycle graph with its line segments. $\mathcal{M}(P)$ is called disconnected if there is no pair i, j with $i \neq j$ and $t_i \succ t_j$.*

Definition 6.2 (Sparse Motorcycle Graphs)**.** *A motorcycle graph is called sparse to a degree k if there are at most k connected motorcycle edges.*

Remark. *A disconnected motorcycle is sparse of degree 1.*

The chance to get a disconnected motorcycle graph for random polygons gets smaller with increasing polygon size. Therefore, polygons with sparse motorcycle graphs of degree 2 were added to increase the sample size.



(a) Disconnected motorcycle graph **(b)** Sparse motorcycle graph (degree 2)

Figure 6.1: Sparse motorcycle graph

Polygon		Overlapping Edges	Insertion Steps			Time $[s]$
n	r		Total	per reflex	per edge	
20	5.38	2.42	5.89	1.09	2.43	0.06
30	9.87	6.39	18.56	1.88	2.90	0.07
40	14.1	8	21	1.49	2.6	0.09
50	18.6	7.2	20.2	1.09	2.81	0.10

Table 1: Sparse motorcycle graphs

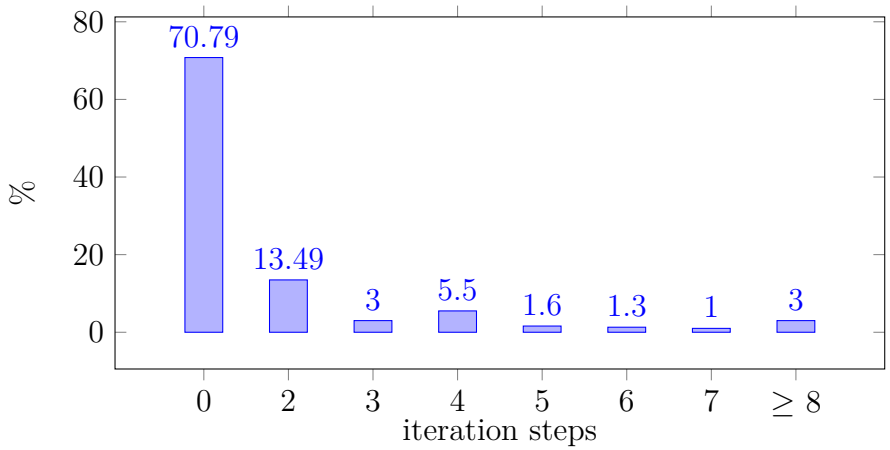


Figure 6.2: Histogram of iteration steps per reflex vertex (over 3000 tested polygons)

Most often (70.79%) there were no overlapping edges at all and therefore no newly created arcs during the algorithm besides the one connected to the reflex vertex. Inserting only one arc is not possible because the algorithm cannot terminate after just one inserted arc and requires at least a second one to finish it.

About 97% of the time a single reflex vertex had less than 8 iteration steps. The highest number of iteration steps for one step was 49. However, different merge orders achieved good results on average for some of the bad examples.

The tested polygons could be calculated efficiently. There are many possible merge orders and they are chosen randomly. However, this did not play a key role for the fast computations. The main reason for it is that these types of polygons either have a small number of reflex vertices with respect to n or that they have a rather thin structure - This means regions do not have many other regions as neighbors and do not have a large influence on other regions as soon as they are merged once.

An extreme example is shown in Figure 6.3 that achieved a high number of iteration steps in one merge step. The reason for the many iteration steps is the high number of overlapping edges in the first step and this step cannot be avoided with another merge order. The high number of overlapping edges are a result of the long but rather thin structure with one motorcycle edge going through the middle of it to the other side of the polygon.

This structure did not occur very often and less than 1% of the tested polygons had a reflex vertex with 20 or more iteration steps and only 0.25% of the reflex vertices had 20 or more steps and most of them had a similar structure.

However, polygons with this structure tend to have either a small average number of steps per reflex or a small number of steps with respect to n because most edges are inserted in a single step. This results in a fast computation for these types of polygons.

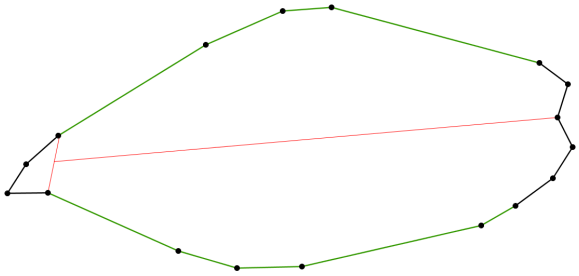


Figure 6.3: Many iteration steps ($n = 18$, overlapping edges in green)

Polygon		**Overlapping**	**Insertion Steps**				**Time** $[s]$
n	**r**		**Total**	**max**	**per reflex**	**per edge**	
17	2	10	40.3	38.3	20.15	4.03	0.08

Table 2: Average parameters for Figure 6.3 from 50 calculations

6.4 Linear Merge Tree

In this section specific polygons with a merge tree of height $\mathcal{O}(n)$ are discussed.

6.4.1 Zipper Motorcycle Graphs

Definition 6.3 (Zipper Motorcycle Graphs)**.** *Let* $Z = \{t_i \mid i \in \{1, \ldots, k\}, k \leq n\}$ *be a connected subgraph of the motorcycle graph with its line segments.* Z *is called zipper motorcycle graph if there exists a permutation* $\pi \in S_k$ *with* $t_{\pi(1)} \succ t_{\pi(2)} > \ldots \succ t_{\pi(k-1)} \succ t_{\pi(k)}$ *and every motorcycle cell is bounded by at most three motorcycle edges of* Z*.*

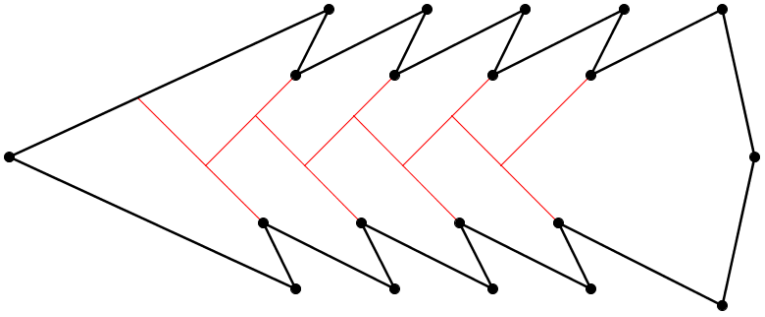


Figure 6.4: Zipper motorcycle graph

Zipper motorcycle graphs can be interpreted as motorcycle graphs with zigzag patterns. There is only a single merge order and the height of the merge tree is $\mathcal{O}(n)$.

Polygon		Overlapping Edges	Insertion Steps			Time $[s]$
n	r		Total	per reflex	per edge	
52	24	23	46	1.92	2	0.27
60	28	27	54	1.93	2	0.33
68	32	31	62	1.94	2	0.35
76	36	35	70	1.94	2	0.44
84	40	39	78	1.95	2	0.49
92	44	43	86	1.95	2	0.55
100	48	47	94	1.96	2	0.66
108	52	51	102	1.96	2	0.69
116	56	55	110	1.96	2	0.71

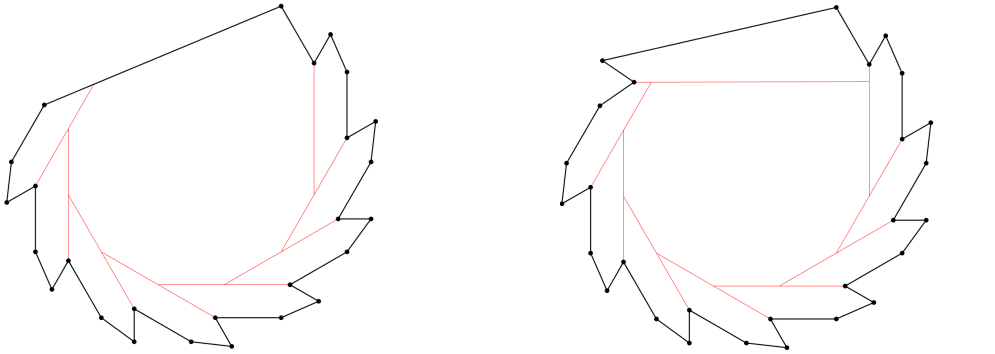
Table 3: Zipper motorcycle graphs (Figure 6.4)

However, the example in Figure 6.4 shows that this can still be very efficient. The number of insertion steps per reflex and per edge is almost constant because the merge process finishes in the first step. This results in a linear running time for this specific polygon and shows that certain polygons with linear height of the merge tree can be computed efficiently.

6.4.2 Pseudo Cycle Motorcycle Graphs

Definition 6.4 (Pseudo Cycle)**.** *Let $M = \{m_i \mid i \in \{1, \ldots, k\}, k \leq n\}$ be a connected subgraph of the motorcycle graph with its line segments. M is called a pseudo cycle if M is not a cycle and there exists a polygon P' and a cycle $C \subset \mathcal{M}(P')$ with $M \subset C$ and $|C| = |M| + 1$.*

Pseudo cycles are structures that are missing one motorcycle edge to be an actual cycle. A pseudo cycle has a fixed merged order to resolve the pseudo cycle.



(a) Pseudo Cycle **(b)** Cycle containing the pseudo cycle

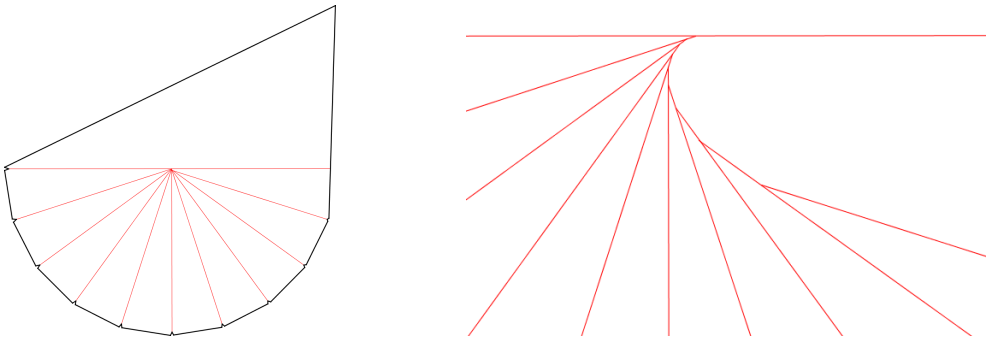
Figure 6.5: Pseudo cycle motorcycle graphs

Polygon		Overlapping Edges	Insertion Steps			Time $[s]$
n	r		Total	per reflex	per edge	
35	11	17	34	3.1	2	0.22
59	19	29	58	3.05	2	0.37
83	27	41	82	3.04	2	0.52
107	35	53	106	3.03	2	0.73
131	43	65	130	3.02	2	0.87
149	49	74	148	3.02	2	1.00
173	57	86	172	3.02	2	1.20

Table 4: Pseudo cycles (Figure 6.5a)

Polygons with pseudo cycles like in Figure 6.5a can be calculated quite efficiently although the depth of the dominance graph is linear and a fixed merge order is given. Furthermore, there was one large region that was involved in every merge step. However, this was still efficient because there were not many overlapping edges per reflex vertex and they didn't have much influence on the large region. This resulted in small local changes and the number of insertion steps was almost constant per reflex vertex and per overlapping edge.

On the other side, pseudo cycles can be quite inefficient as well. The idea is to keep the region that is connected to all motorcycle edges rather small so that a new region overlaps as many of the already merged regions as possible.



(a) Pseudo cycle **(b)** Zoom of the central region

Figure 6.6: Pseudo cycles with small central region

Polygon		Overlapping Edges	Insertion Steps			Time $[s]$
n	r		Total	per reflex	per edge	
31	10	15	61	6.1	4.07	0.22
37	12	19	85	7.08	4.47	0.26
43	14	23	114	8.14	4.97	0.32
49	16	26	143	8.94	5.5	0.39
55	18	29	176	9.78	6.07	0.43
61	20	32	212	10.60	6.625	0.51
67	22	36	254	11.45	7.06	0.56
211	70	121	212	10.60	6.625	2.99
241	80	140	3019	37.74	21.56	3.71
271	90	157	3806	42.29	24.24	4.47
301	100	177	4692	46.92	26.51	5.5
451	150	276	10531	70.21	38.16	10.80

Table 5: Pseudo cycles (Figure 6.6a)

The average number of steps per reflex vertex and overlapping edge grow for larger polygons which was not the case for the pseudo cycles in Table 4. In addition to the average numbers, the amount of steps for each reflex vertex are of interest as well. The typical behavior of these specific polygons is shown in Figure 6.7 in the case of a polygon with 100 reflex vertices.

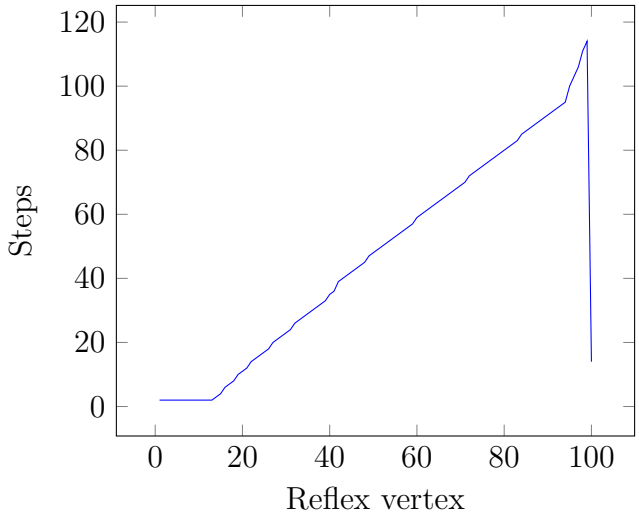


Figure 6.7: Pseudo cycle ($n = 301, r = 100$)

In the first few steps there are usually not many overlaps because the reflex vertices start with a rather low speed compared to the remaining reflex vertices. After the first few reflex vertices are inserted the newly inserted edges start to influence all or almost all of the already inserted regions. The number of steps and created arcs grows linearly until the last few vertices.

The first reflex vertices do not cause many overlaps and the last reflex vertices are the ones that overlap the first few regions the most. This gives a few extra steps at the end. The drop at the end is caused by the last vertex that merges the half circle structure at the bottom with the top region.
This linear growth can also be seen in Figure 6.8 where the maximum number of steps for different amounts of reflex vertices is shown.

The polygons are created with r reflex vertices and $n = 3r + 1$ vertices. Therefore, the dominance graph has a depth of $\Theta(n)$ and each merge step (except the first few) has linearly many created arcs which results in a runtime of $\Omega(n^2)$ for this type of polygon.

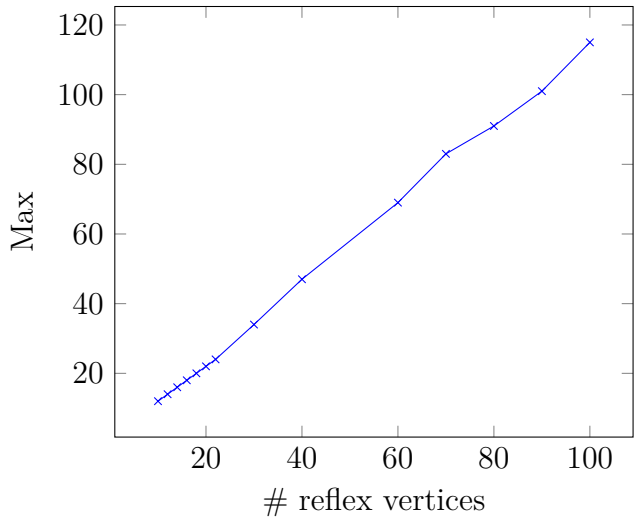


Figure 6.8: Maximum number of steps for different numbers of reflex vertices

6.4.3 General Polygons

In this section more general polygons are tested. These include several different types and structures of polygons and motorcycle graphs.

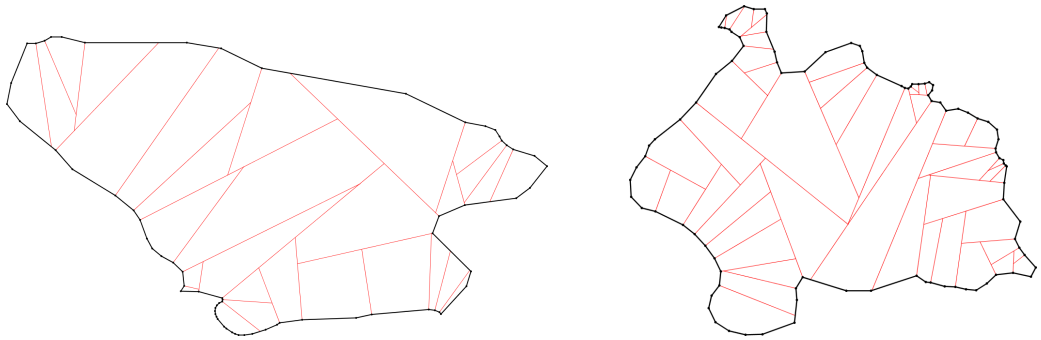


Figure 6.9: Examples of tested polygons

Polygon		Overlapping Edges	Insertion Steps			Time $[s]$
n	r		Total	per reflex	per edge	
20	6.23	4.10	10.05	1.61	2.45	0.05
30	10.81	6.37	15.64	1.44	2.45	0.09
40	15.97	8.87	21.51	1.35	2.42	0.22
50	19.56	13.08	31.46	1.61	2.40	0.37
60	25.47	17.23	44.82	1.76	2.60	0.52
70	30.43	20.16	49.46	1.62	2.45	0.64
80	34.88	22.49	56.68	1.82	2.52	0.75

Table 6: General polygons

The experimental evaluation of these polygons show that the algorithm works quite efficiently for them. The average number of iterations per reflex vertex and per overlapping edge stayed roughly constant.

The maximum number of iteration steps was 43 achieved by a polygon with 28 reflex vertices. However, almost half of the reflex vertices had 0 iteration steps. All polygons with 10 or more reflex vertices had less than 5 iterations steps per reflex vertex on average. High averages are caused by structures like the one in Figure 6.3. Therefore, even extreme cases like polygons with high maximum of iteration steps and high averages of iteration steps could be computed efficiently.

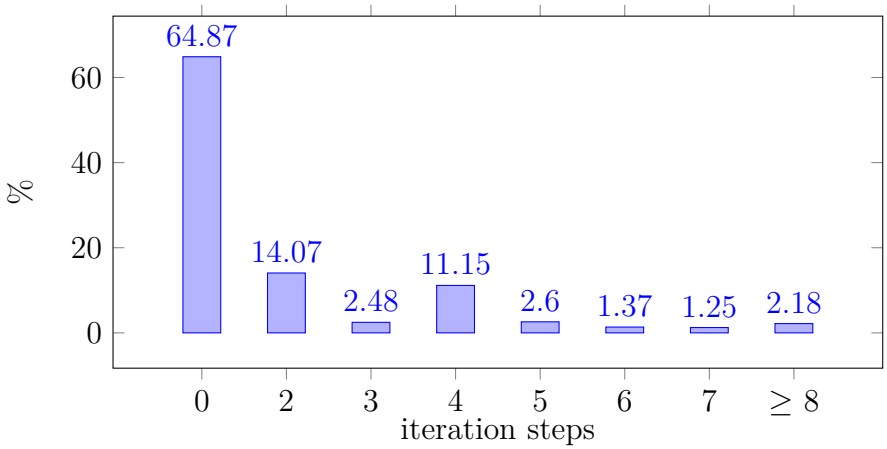


Figure 6.10: Histogram of iteration steps per reflex vertex (over 4000 tested polygons)

The histogram in Figure B.10 reflects this behavior and shows similar results like the histogram for sparse motorcycle graphs (Figure 6.2).

There is a spike at 4 iterations. Three iteration steps per reflex vertex are only possible if there are no newly created arcs on one side and exactly three on the other side. On the other hand, four iteration steps have the additional combination of the common two iterations on each side.

7 Conclusion

The main goal in this thesis was to present a new algorithm for computing the straight skeleton that is useful for practical purposes and also easy to implement. For this, an implementation of this algorithm was provided and used to study the algorithm.

In 2014 Cheng, Mencel and Vigneron [10] presented their algorithm for computing the straight skeleton of a polygon. Their algorithm runs in $\mathcal{O}(n \log(n) \log(r) + r^{\frac{4}{3}+\varepsilon})$ time, which is currently the best known theoretical running time. However, this algorithm is quite involved and they did not provide an implementation to show its practical usefulness.

The algorithm presented in this thesis has an expected running time of $\mathcal{O}(d\, n \log(n))$ where d is the height of the merge tree. In the best case, the merge tree has a height of $\mathcal{O}(\log(n))$ which results in an expected running time of $\mathcal{O}(n \log^2(n))$. In the worst case, the height is linear and this results in an expected running time of $\mathcal{O}(n^2 \log(n))$ for computing the straight skeleton of a simple polygon.

The experimental evaluation showed that the algorithm works very well in practice. The algorithm was shown to be slow for certain types of polygons (pseudo cycles) where the merge tree has linear height and each newly merged reflex vertex influences almost all previously added regions. However, this example was specifically constructed and not a single one of the general polygons showed a similar behavior.

Outlook and further work

There are several open questions related to the presented algorithm. At the moment, the algorithm and the implementation only work for simple polygons without holes.

Furthermore, the merge order (merge tree) was chosen randomly and the algorithm was tested multiple times to find an average running time. The open problem is to find an efficient merge order. It is in general desirable to find a merge tree with balanced height, but for practical purposes a linear height of the merge tree can work very well and it is currently unknown to decide a priori which merge tree results in the optimal running time.

In addition, an open problem are possible optimizations for non-balanced merge trees. For example, pseudo cycles that have merge trees with linear height can be approached differently. Cycles can be merged efficiently due to a balanced binary merge approach. This can be used as an idea to merge pseudo cycles as well because they almost behave likes cycles. The underlying problem is the resolving of dominant motorcycle edges which lead to problems when the algorithm is used

without modifications.

There is a multitude of different polygons and structures of the motorcycle graph. Only a handful of these could be analyzed within the scope of this thesis. However, the algorithm and the program can be used to study more polygons for practical purposes. The tested polygons included general polygons of many different sizes and structures to study general behavior. Furthermore, a few selected polygons and structures were chosen to showcase good and bad behavior of the algorithm. However, it is still of interest to classifiy polygons with good and bad behavior and find more examples.

The pseudo cycles showed that polygons with identical dominance graphs and identical merge trees can behave vastly different. Therefore, it might be interesting to study how the structure of the polygon affects the algorithm in general and for identical dominance graphs. For this, the size and position of motorcycle cells are especially interesting. Small central cells might have a bad influence because many other regions can overlap into this cell which can result in a lot of iteration steps.

A Examples

This short chapter presents a few selected straight skeletons which are calculated by using the implementation presented in Chapter 5. These examples include straight skeletons of polygons that were mentioned in this thesis.

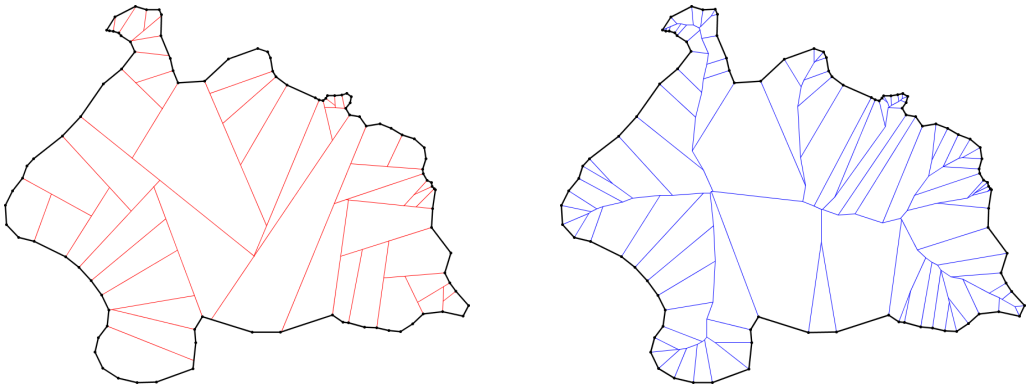


Figure A.1: Straight skeleton of polygon with 99 vertices

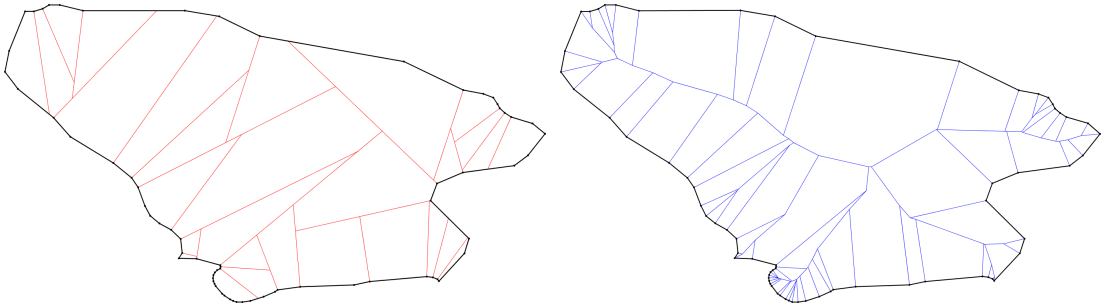


Figure A.2: Straight skeleton of polygon with 69 vertices

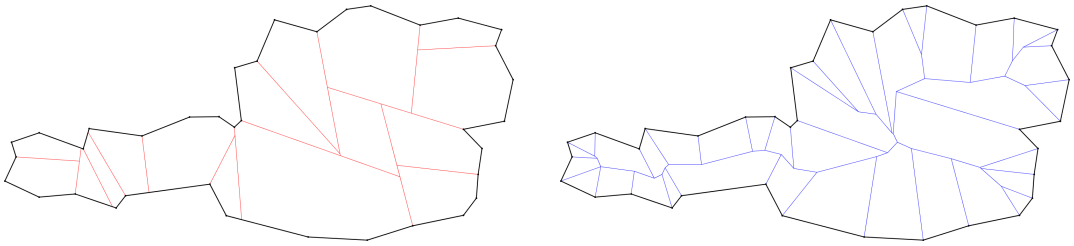


Figure A.3: Polygon in the shape of Austria containing a cycle

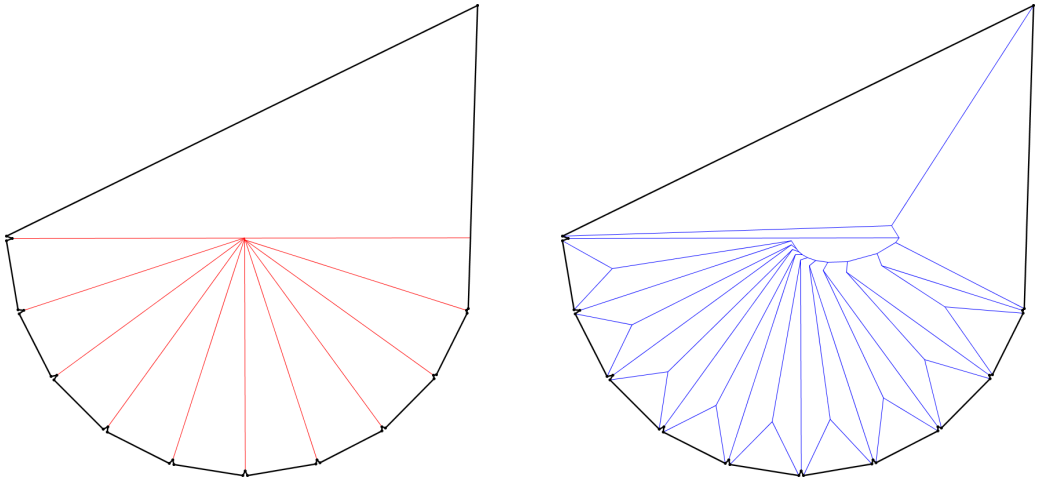


Figure A.4: Ineffcient pseudo cycle

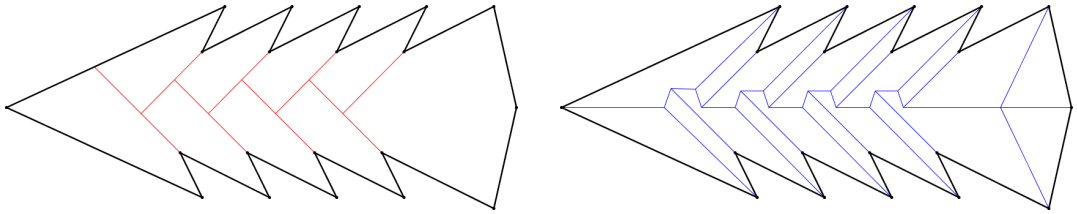


Figure A.5: Zipper motorcycle graph

B Evaluation

B.1 Bouncing Reflex

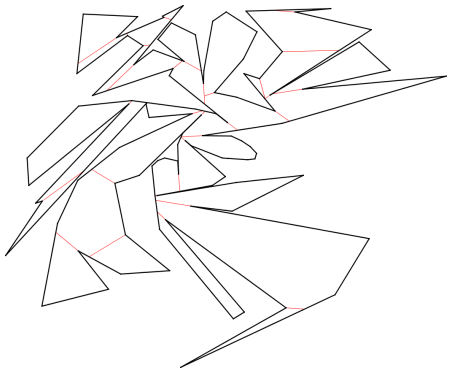


Figure B.1: Bouncing reflex [5]

# Polygons	Polygon n	Polygon r	Overlapping	Insertion Steps Total	Insertion Steps per reflex	Insertion Steps per edge	Time $[s]$
64	101	39.30	9.62	21.23	0.54	2.21	0.15
54	105	35.62	9.36	20.05	0.56	2.14	0.14

Table 7

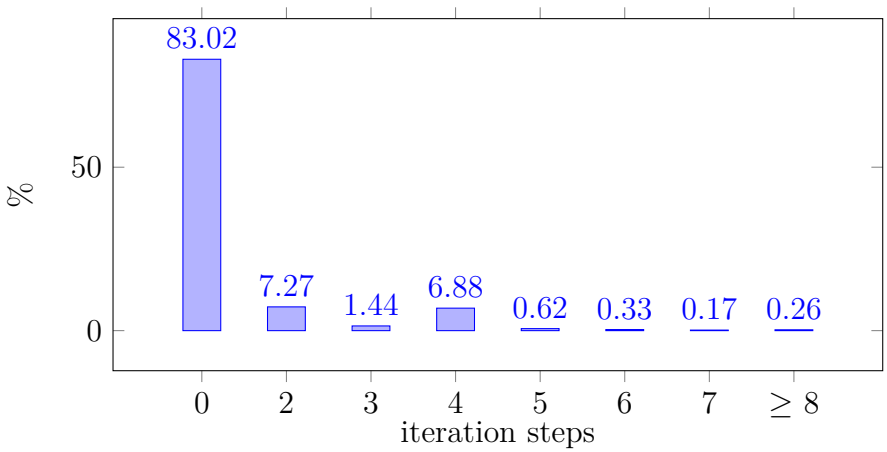


Figure B.2: Histogram of iteration steps per reflex vertex (118 tested polygons)

Remark. *The highest number of iteration steps for a single reflex vertex was 14. The highest average value for iterations per reflex was* 1.55 *($r \geq 3$).*

B.2 Convex Bottom

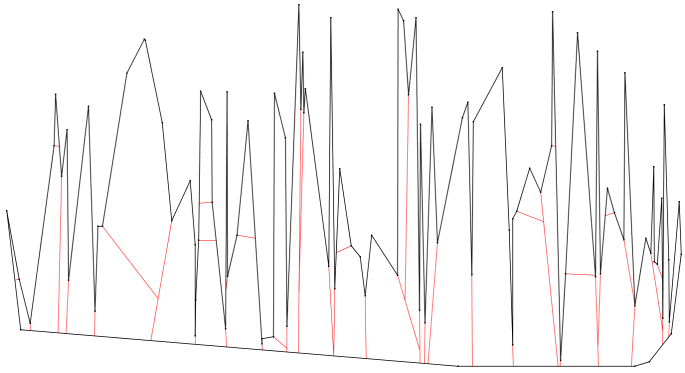


Figure B.3: Convex bottom [5][6][19]

# Polygons	Polygon n	Polygon r	Overl.	Insertion Steps Total	Insertion Steps per reflex	Insertion Steps per edge	Time [s]
62	100-119	50.92	12.94	32.36	0.64	2.51	0.24
48	120-139	59.88	14.88	38.13	0.64	2.56	0.27
49	140-159	70.73	17.33	44.51	0.63	2.57	0.35
41	160-179	79.87	19.24	49.63	0.62	2.58	0.38
50	180-200	91.22	21.63	55.39	0.61	2.56	0.44

Table 8

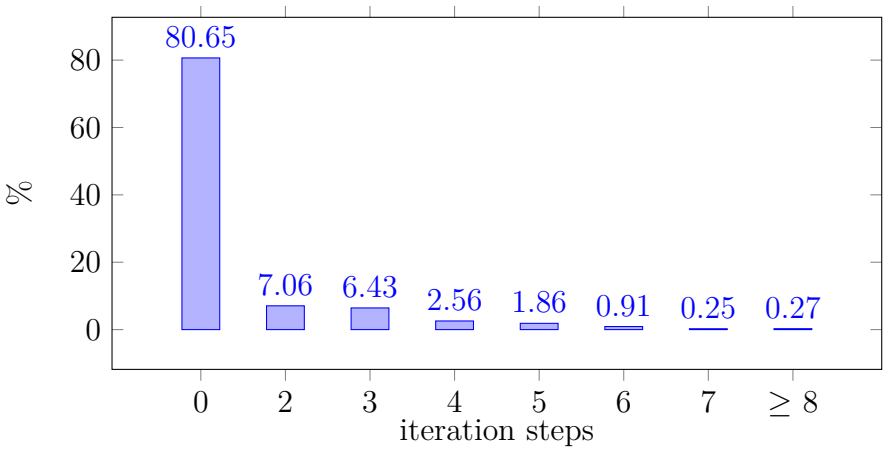


Figure B.4: Histogram of iteration steps per reflex vertex

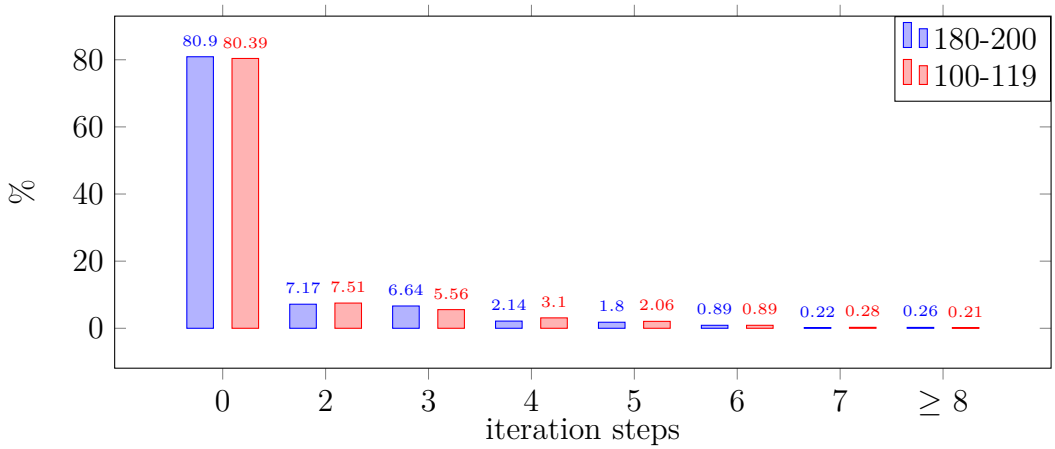


Figure B.5: Comparison of histograms

Remark. *The highest number of iteration steps for a single reflex vertex was 15. The highest average value for iterations per reflex was* 1.27*. However, more than 95% of the tested polygons had an average below 1.*

B.3 Space Partitioning

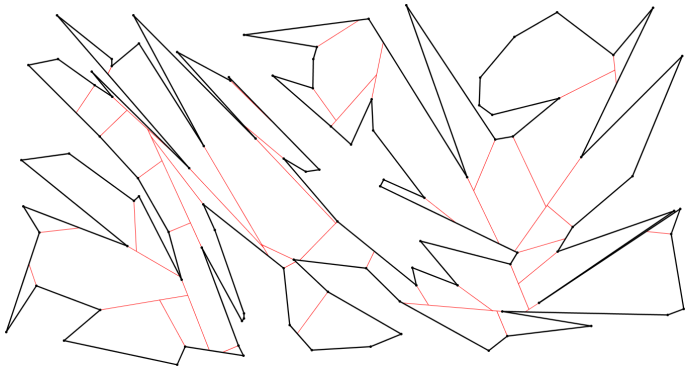


Figure B.6: Space partitioning [5][6][19]

# Polygons	Polygon n	Polygon r	Overl.	Insertion Steps Total	Insertion Steps per reflex	Insertion Steps per edge	Time $[s]$
29	100-119	52.00	26.59	60.59	1.17	2.28	0.37
30	120-139	61.80	31.41	73.25	1.19	2.33	0.45
55	140-159	72.73	33.25	77.15	1.06	2.32	0.53
50	160-179	81.18	35.43	80.05	0.99	2.26	0.63
37	180-200	92.73	42.80	99.95	1.08	2.34	0.72

Table 9

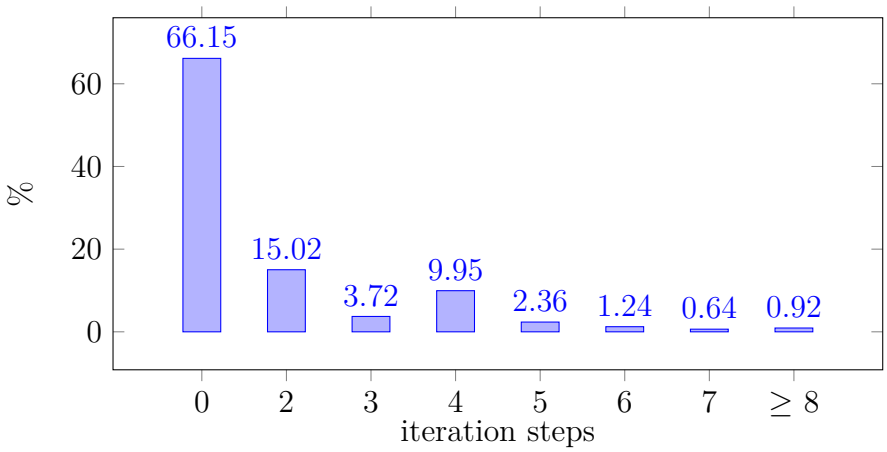


Figure B.7: Histogram of iteration steps per reflex vertex

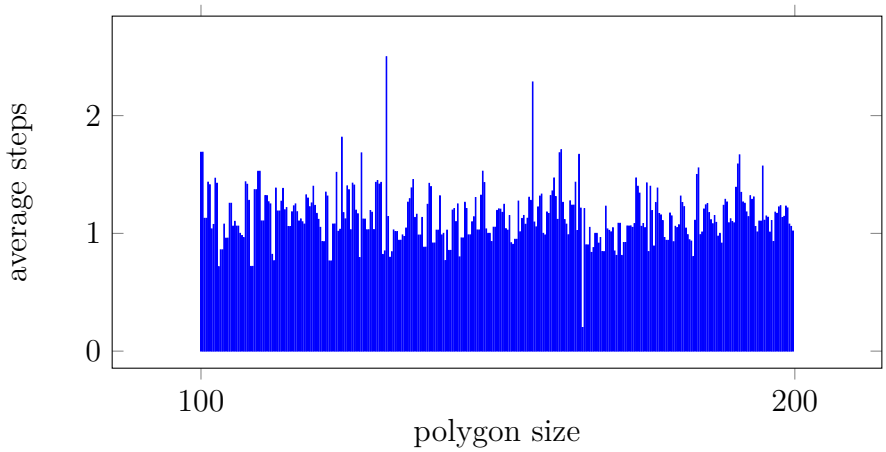


Figure B.8: Average number of steps per reflex vertex for increasing polygon sizes

Remark. *The highest number of iteration steps for a single reflex vertex was 19. The highest average value for iterations per reflex was* 2.50. *However, only two of the tested polygons had an average above 2.*

B.4 Steady Growth

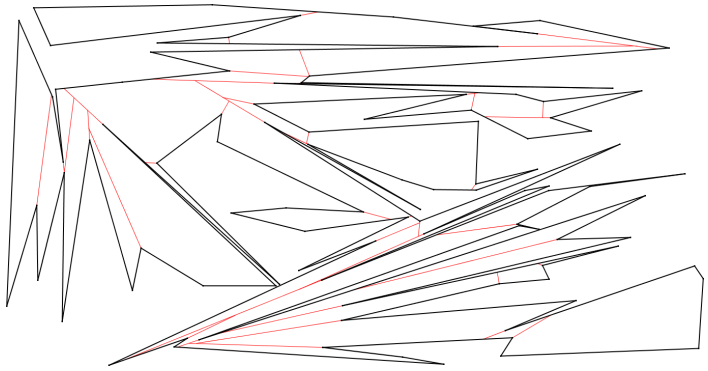


Figure B.9: Steady growth [5][6][19]

# Polygons	Polygon n	Polygon r	Overl.	Insertion Steps Total	Insertion Steps per reflex	Insertion Steps per edge	Time [s]
43	100-119	50.30	20.64	47.56	0.95	2.30	0.32
44	120-139	59.07	23.51	54.20	0.92	2.31	0.45
37	140-159	68.46	25.02	57.16	0.83	2.28	0.56
37	160-179	78.24	30.84	71.09	0.91	2.30	0.74
35	180-189	85.53	31.19	70.26	0.82	2.25	0.82
39	190-200	90.21	31.08	71.96	0.80	2.32	0.83

Table 10

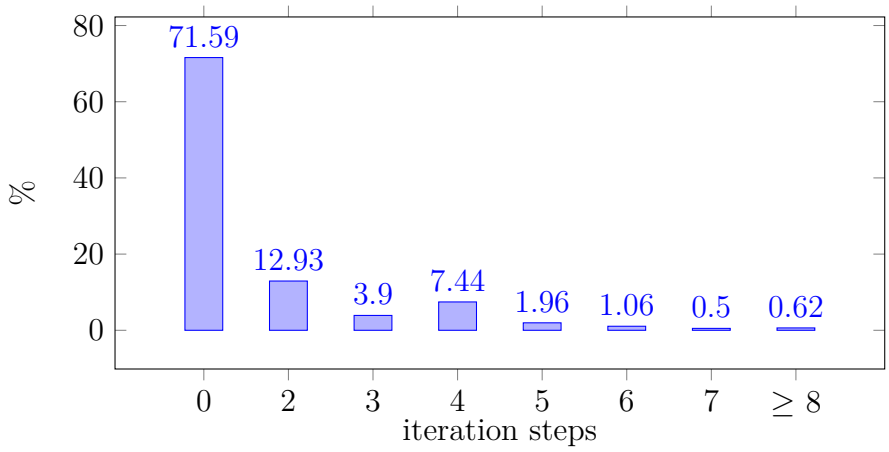


Figure B.10: Histogram of iteration steps per reflex vertex

Bibliography

[1] Franz Aurenhammer, Michael Steinkogler. *On Merging Straight Skeletons*. EuroCG, Berlin, 2018.

[2] Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. *A Novel Type of Skeleton for Polygons*. Journal of Universal Computer Scienece, vol. 1, no. 12, pages 752-761, 1995.

[3] Oswin Aichholzer, Franz Aurenhammer. *Straight Skeletons for General Polygonal Figures in the Plane*. In Computing and Combinatorics, volume 1090 of Lecture Notes in Computer Science, pages 117–126. Springer, 1996.

[4] Gernot Walzl. *Straight Skeletons - From Plane to Space* (PhD thesis). Institute for Theoretical Computer Science, University of Technology, Graz, Austria, 2015.

[5] Thomas Auer. *Heuristics for the Generation of Random Polygons* (Master's thesis). Faculty of Natural Science, University of Salzburg, Austria, 1996.

[6] T. Auer and M. Held. *RPG: Heuristics for the Generation of Random Polygons*. In Proc. 8th Canadian Conference Computational Geometry, 1996, pp. 38-44.

[7] F. Aurenhammer. *Weighted skeletons and fixed-share decomposition*. Computational Geometry, 40(2), pp.93-101., 2008.

[8] Siu-Wing Cheng and Antoine Vigneron *Motorcycle Graphs and Straight Skeletons*. Algorithmica, 47(2), pp.159-182., 2002.

[9] Antoine Vigneron and Lie Yan. *A Faster Algorithm for Computing Motorcycle Graphs*. Discrete & Computational Geometry, 52(3), pp.492-514, 2014.

[10] Cheng, S., Mencel, L. and Vigneron, A. *A Faster Algorithm for Computing Straight Skeletons*. ACM Transactions on Algorithms, 12(3), pp.1-21., 2016.

[11] Aurenhammer, F. and Walzl, G. *Straight Skeletons and Mitered Offsets of Nonconvex Polytopes*. Discrete & Computational Geometry, 56(3), pp.743-801., 2016.

[12] L Paul Chew. *Building voronoi diagrams for convex polygons in linear expected time*. Technical report, 1990.

[13] Chin, F., Snoeyink, J. and Wang, C. *Finding the Medial Axis of a Simple Polygon in Linear Time*. Discrete & Computational Geometry, 21(3), pp.405-420., 1999.

[14] Alok Aggarwal, Leonidas Guibas, James Saxe, and Peter Shor. *A linear time algorithm for computing the voronoi diagram of a convex polygon.* In Proceedings of the nineteenth annual ACM symposium on Theory of computing, STOC '87, pages 39–45, New York, NY, USA, 1987. ACM.

[15] W. Mann, M. Held, S. Huber. *Computing Motorcycle Graphs Based on Kinetic Triangulations*. Proc. 24th Canadian Conf. on Computational Geometry, p. 187-192, Charlottetown, P.E.I., Canada, Aug 2012.

[16] Bederson, B. B., Grosjean, J. and Meyer, J. *Toolkit Design for Interactive Structured Graphics*, IEEE Transactions on Software Engineering, 30 (8), pp.535-546., 2004

[17] Web.archive.org. (2018). *Piccolo2D - A Structured 2D Graphics Framework*. [online] Available at: `https://web.archive.org/web/20180316072624/http://piccolo2d.org` [Accessed 16 Mar. 2018].

[18] Web.archive.org. (2018). GeoGebra - Dynamic Mathematics. [online] Available at: `https://web.archive.org/web/20180415015024/https://www.geogebra.org/home` [Accessed 15 Apr. 2018].

[19] Web.archive.org. (2018). *Randomly generated polygons* [online] Available at: `https://web.archive.org/web/20170710054754/http://web.informatik.uni-bonn.de/I/GeomLab/RandomPolygon/index.html.en` [Accessed 27 Apr. 2018].

[20] Weisstein, Eric W. *Polygon Area.* From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/PolygonArea.html

[21] Beyer, W. H. (Ed.). *CRC Standard Mathematical Tables, 28th ed.* Boca Raton, FL: CRC Press, pp. 123-124, 1987.

[22] D. Eppstein, J. Erickson *Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions*. Discrete Comput. Geom., 22 (4) (1999), pp. 569-592.

[23] Erik D. Demaine, Martin L. Demaine, and Anna Lubiw. *Folding and cutting paper*. In Post-Conf. Proc. of the Japan Conference on Discrete and Computational Geometry (JCDCG 1998), volume 1763 of Lecture Notes in Computer Science, pages 104–118. Springer, 1998.

[24] E. D. Demaine, M. L. Demaine, J. F. Lindy, and D. L. Souvaine. *Hinged dissection of polypolyhedra*. In Proceedings of the 9th Workshop on Algorithms and Data Structures (WADS 2005), volume 3608 of Lecture Notes in

Computer Science, pages 205–217, Waterloo, Ontario, Canada, August 15–17 2005.

[25] Gill Barequet, David Eppstein, Michael Goodrich, and Amir Vaxman. *Straight skeletons of three-dimensional polyhedra.* In Algorithms - ESA, volume 5193 of Lecture Notes in Computer Science. Springer, 2008.

[26] R. Laycock,A. Day. *Automatically generating large urban environments based on the footprint data of buildings.* In: Proceedings of 8th ACM Symposium on Solid Modeling and Applications, pp. 346–351 (2003)

[27] T. Kelly, P. Wonka. *Interactive architectural modeling with procedural extrusions.* ACMTrans. Graph. 30(2), 14:1–14:15 (2011)

[28] Jan-Henrik Haunert and Monika Sester. *Area collapse and road centerlines based on straight skeletons.* Geoinformatica, 12(2):169–191, 2008.